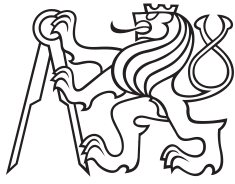


Bachelor Project



**Czech
Technical
University
in Prague**

F3

**Faculty of Electrical Engineering
Department of Cybernetics**

Personal Life Management System

Jan Daniel

**Supervisor: Ing. Božena Mannová, Ph.D.
Study program: Open Informatics
Specialization: Artificial Intelligence and Computer Science
May 2023**

I. Personal and study details

Student's name: **Daniel Jan** Personal ID number: **495553**
Faculty / Institute: **Faculty of Electrical Engineering**
Department / Institute: **Department of Cybernetics**
Study program: **Open Informatics**
Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Personal Life Management System

Bachelor's thesis title in Czech:

Systém pro správu osobního života

Guidelines:

1. Get to know the issue of personal development, which focuses on increasing self-awareness, tracking life goals, personal challenges and long-term aspirations.
2. Analyze existing applications available to you and evaluate them.
3. Based on the analysis, design the basic functionalities of the proposed application, such as saving reminders, sending event notifications, managing personal finances, etc. and supplement them with monitoring personal development.
4. When designing, focus on the use of AI in tracking personal development, such as the possibility of predicting the user's chance of maintaining existing habits based on their previous activity.
5. Choose the application architecture and technology for implementation.
6. Design a friendly user experience. Also focus on the issue of secure communication.
7. Implement and test the application.
8. Evaluate the results and suggest any additional functionality or other improvements.
9. Use appropriate means of software engineering when solving.

Bibliography / sources:

- [1] Roger S. Pressmann Bruce Maxim: Software Engineering: A Practitioner's Approach , ISBN-10: 9780078022128
- [2] <https://monday.com/blog/project-management/project-management-software-for-individuals/>
- [3] <https://www.proprofsproject.com/blog/personal-task-management-tools/>
- [4] <https://timelyapp.com/blog/best-task-management-app>

Name and workplace of bachelor's thesis supervisor:

Ing. Božena Mannová, Ph.D. Center for Software Training FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **13.02.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

Ing. Božena Mannová, Ph.D.
Supervisor's signature

prof. Ing. Tomáš Svoboda, Ph.D.
Head of department's signature

prof. Mgr. Petr Páta, Ph.D.
Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Student's signature

Acknowledgements

I would like to thank my supervisor Ing. Božena Mannová, Ph.D. for her counsel and guidance throughout the process of writing this theses.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 26.5.2023

Jan Daniel

Abstract

The work is focused on the issue of personal development, which focuses on increasing self-awareness, tracking life goals, personal challenges and long-term aspirations. As part of this work, an analysis of existing applications and their evaluation was carried out. Based on the analysis, the architecture of the application was designed and its design was carried out. The system was implemented and testing, including user tests, was carried out. The work also deals with the possibilities of using machine learning methods for this type of application.

Keywords: Personal development, Personal development application, Personal development system, Django, Machine learning usage

Supervisor: Ing. Božena Mannová,
Ph.D.
Study program: Open Informatics
Specialization: Artificial Intelligence and
Computer Science
Department of Computer Science,
Karlovo náměstí 13,
Praha 2

Abstrakt

Práce je zaměřena na problematiku osobního rozvoje, který se zaměřuje na zvýšení sebeuvědomění, sledování životních cílů, osobních výzev a dlouhodobých aspirací. V rámci této práce byla provedena analýza existujících aplikací a jejich vyhodnocení. Na základě provedené analýzy byla navržena architektura aplikace a proveden její návrh. Systém byl implementován a bylo provedeno testování včetně uživatelských testů, Práce se zabývá i možnostmi využití metod strojového učení pro tento typ aplikace.

Klíčová slova: Osobní rozvoj, Aplikace osobního rozvoje, Django, Systém pro správu osobního rozvoje, Využití strojového učení

Překlad názvu: Aplikace pro správu osobního života

Contents

1 Introduction	1	4.4.2 Registered User wants to log in	20
1.1 My motivation	1	4.4.3 Registered user wants to log out	20
1.2 Subject matter	2	4.4.4 Logged in user wants to change their personal information	20
2 Motivation as a concept and research behind habit building	3	4.4.5 Logged in user wants to delete their profile	21
2.1 Motivation	3	4.4.6 Logged in user wants to see an overview of their activities	21
2.2 Inspiring intrinsic motivation	3	4.4.7 Logged in user wants to see some upcoming event displayed on their dashboard	21
2.3 Justification of the leading role of habits	4	4.4.8 Logged in user wants to see specific habit/ToDo	21
2.4 About habits	4	4.4.9 Logged in user wants to create an activity	21
2.5 How to build a habit	5	4.4.10 Logged in user wants to see their statistics and analysis of his current performance and behaviour	22
2.6 Plugging habits into motivation	7	4.4.11 Logged in user wants to see their schedule	22
2.7 Conclusion	7	4.4.12 Logged in user wants to add a custom event to their calendar	22
3 A survey of relevant existing solutions	8	5 Technological stack and architecture proposal	23
3.1 List of existing application and research methodology	8	5.1 Basic software application architectural patterns	23
3.2 Habitica	9	5.1.1 Model View Controller (MVC)	23
3.3 Way of life	10	5.1.2 Model View Presenter(MVP)	23
3.4 Habitify	11	5.2 Server side Web application frameworks	24
3.5 Productive	12	5.2.1 Django	24
3.6 Summary and conclusion	13	5.2.2 Spring Boot	25
4 Requirements and use cases	15	5.3 Database	25
4.1 General concepts	15	5.3.1 MySQL vs PostgreSQL	25
4.2 General functionality draft	16	5.4 Frontend frameworks	26
4.2.1 User	16	5.4.1 Vue JS	26
4.2.2 Habit	17	5.4.2 React JS	26
4.2.3 ToDo	17	5.5 CSS Frameworks	27
4.2.4 Calendar	17	5.5.1 Bulma.io	27
4.2.5 Statistics section	17	5.5.2 Bootstrap	27
4.3 Functional requirements	18	5.6 Technologies selected for the implementation	28
4.3.1 Landing page	18	5.6.1 Web application framework	28
4.3.2 Registration	18	5.6.2 Database	29
4.3.3 Login	18	5.6.3 Frontend framework	29
4.3.4 User profile	18		
4.3.5 User dashboard	18		
4.3.6 Habit	19		
4.3.7 Habit list	19		
4.3.8 Todos	19		
4.3.9 Todo list	19		
4.3.10 Calendar	19		
4.3.11 Calendar events	19		
4.3.12 Statistics section	20		
4.4 Use cases	20		
4.4.1 User wants to register	20		

5.6.4 CSS Framework	29	9.2 Usage & benefits	69
5.6.5 Database model draft	30	9.3 Possible functionality extension & upgrades	69
6 Implementation	33	9.3.1 Messaging inside the application	69
6.0.1 Application modules	33	9.3.2 An internal social network ..	70
6.0.2 Environment preparation . . .	33	9.3.3 Achievements & streaks	71
6.0.3 Helper structures	34	9.3.4 Public API	71
6.0.4 Codebooks	35	9.3.5 Extend habit type range	71
6.1 Backend development & API . . .	36	9.3.6 UI facelift	71
6.1.1 Login & Registration	36	9.3.7 Integration with smart watch	72
6.1.2 Habits	38	9.3.8 Finishing the implementation of other modules	72
6.1.3 ToDos	41	A Project structure	74
6.1.4 Users Profile	42	B Application visual appearance	78
6.2 User interface & Frontend	43	Bibliography	84
6.2.1 Landing page	43		
6.2.2 Habits page	44		
6.2.3 ToDo page	45		
6.2.4 Database model	46		
6.3 Experimental scheduler feature .	49		
6.3.1 About Celery	49		
6.3.2 Installation	49		
6.3.3 Implementation	49		
6.3.4 Running	50		
6.3.5 Tasks implementation	50		
6.4 Next steps: Modules for future development	51		
6.4.1 Dashboard	51		
6.4.2 Statistics section	52		
6.4.3 Backend & API	53		
7 Testing	54		
7.1 Automated testing	54		
7.1.1 Example - Testing Delete Habit Endpoint	55		
7.2 Usability testing	57		
7.2.1 Testing scenario 1	57		
7.2.2 Testing scenario 2	59		
8 Empowering the application with Machine Learning	61		
8.1 Problem definition	61		
8.2 Assumptions	62		
8.3 Possible problem formalizations.	63		
8.3.1 Supervised learning approach	63		
8.4 Goal and advantages	67		
9 Conclusion	68		
9.1 Looking back: An assessment of the process	68		

Figures

5.1 Database Schema	31
6.1 Example of a helper structure usage in an API endpoint	34
6.2 Example of a Helper structure implementation	35
6.3 Codebook database tables	36
6.4 Login API endpoint	37
6.5 API endpoint for registering . . .	38
6.6 The central endpoint for loading of the Habit page	39
6.7 The endpoint for creating a Habit entity	40
6.8 API endpoint for editing the Habit plan.	41
6.9 Deletion of Habit and all related records	42
6.10 Login landing page UI	43
6.11 Registration page UI	44
6.12 Habits page design draft	45
6.13 Authentication logic database schema	47
6.14 Habit database tables schema .	48
6.15 The code responsible for sending emails with reminders to the users	51
7.1 Setup method of the test to prepare necessary data for the testing	56
B.1 Login page of the application. . .	78
B.2 Registration page of the application.	79
B.3 Habits module page.	79
B.4 Habits module page modal for creating new Habit.	80
B.5 Habits module page modal for adding a new Habit session.	81
B.6 Habits module page modal for deleting a Habit	81
B.7 Todo module page.	82
B.8 Todo module page modal for creating a new Todo.	83

Tables

3.1 Existing solutions comparison . .	13
---------------------------------------	----



Chapter 1

Introduction



1.1 My motivation

I have chosen the personal development domain because the constant conflict between procrastination and the will to do something with myself, better manage my time and develop in any and all aspects of life is very close to me. Through my life I have tried countless apps for building habits, keeping those which are beneficial and getting rid of those which are unproductive or even harmful. But unfortunately none of them seemed very fitting for me and it always ended up the same - after a few weeks I forgot that I had such app and about the habits I wanted to develop and fell right back into my old routines. It is undeniable that in order to build new habits, manage ones time and overall grow as a person one needs a lot of self discipline and a strong willpower, however I believe, that self discipline and willpower alone are not enough for most people. Motivation fades with excitement and so does willpower. In my opinion in order to build an application which can really help people with building new habits and organising both their time and life one must not only consider a human psyche in the application design, but the application itself must be based on a research and facts regarding human psychology of habits, motivation and why people behave the way they behave and it must go along the way of the reasons there even are habits and not against it.

■ 1.2 Subject matter

Self development applications, or so called "Habit trackers" are applications which, clearly, focus mostly on building habits. These applications usually work in a way, that they let one clearly and in detail define the habits they are either trying to develop, keep or get rid of. They let one define time and place where the activity should take place, remind them that they should be getting ready to go running, go to the gym or drink water for an instance. Most of the times they split certain activities one wants to perform into specific categories such as habits, goals, tasks, todo's and many more. There will be a discussion of the result of a research regarding other, more colorful and fruitful, methods of motivating people and helping them stick with their desired habits in further chapters.

Chapter 2

Motivation as a concept and research behind habit building

2.1 Motivation

Motivation has been a central concern of psychologists for decades. Other than it being at core of biological, cognitive and social regulation it is interesting because of its real world every day consequences. According to this paper(1) motivation is not a singular but rather a consequence of many internal and external factors. There are two main types of motivation - intrinsic and extrinsic.

Intrinsic motivation - According to self determination theory (1) intrinsic motivation is what is inadvertently rooted in every person from youth. According to a paper (2) it is defined as the doing of some activity not for the subsequent rewards but rather for the inherent satisfaction of performing it. It has been observed on many animals but most notably on humans. It is best visible on infants or small children when exploring their surroundings - tasting, touching and grasping various objects around them does not bring them any benefits, but they are doing it for the satisfaction of the exploring itself. It is perhaps best shown in contrast with extrinsic motivation.

Extrinsic motivation - On the contrary to intrinsically motivated actions, extrinsically motivated actions are done solely for some sort of outcome(3). It may be not getting punished for not doing something or it can be getting a reward for doing it.

According to this paper(1) studies have shown that when some activity is intrinsically motivated people tend to be more focused, have better confidence, persistence and more creativity.

2.2 Inspiring intrinsic motivation

The authors of Self Determination Theory (1) state, that intrinsic motivation is present from birth in all people and it can be supported to grow or diminished by external factors and the environment. According to (1) positive feedback serves well in promoting intrinsic motivation and along with

communication can inspire feelings of competence which have been observed to enhance intrinsic motivation. However the feeling of competence alone is not enough, when it is not accompanied by feelings of autonomy and self determination.

According to the authors of (1) one must feel related to the activity, competence to perform it, autonomy (Here autonomy does not mean necessarily doing it alone, but rather believing in ones ability to perform the task competently) and understanding why are they performing the activity in order to be intrinsically motivated. The authors then offer an example of children who have been led to do some task, which they did not at the time understand the reason of, and did not have the tools to perform it adequately and so have subsequently shown low internalization of the activity as well as resentment for it in the future. In summary one should know what are they doing, why are they doing it and its benefits, be confident in their ability to perform, feel competent to do so and most importantly the activity should be aligned with their beliefs and better yet be a part of their identity.

■ 2.3 Justification of the leading role of habits

In order for an application to help people succeed in creating new habits and become a better person, it needs to go with the natural flow of how and why the human mind creates habits and not against it. The main issue with many similar applications is that they let people track their progress, define their goals, organise their life but they don't address the core issue why most people fail which is that they tend to have strong start since they are motivated to change something, but after few weeks, once the motivation fades away, they do not have the energy or the willpower to force themselves to do the activity, to which they have no connection other than that they want to do it since they have been forcing themselves all that time. However if a person develops affection for the activity and makes it part of their personality, or rather changes their personality into someone for whom this activity is aligned with their beliefs and values, then they can truly succeed, because it is part of them. This is what is called identity based habits as opposed to outcome based habits.

■ 2.4 About habits

The reason that habits even exist is simple - energy conservation. Human mind is trying to optimize everything in order to preserve energy and so if it notices that it is doing one or multiple actions repeatedly in the same order and same environment at a similar time it tries to automate(4) that behaviour so it can be done subconsciously without even thinking about the task at hand.

Every person at some point notices that they are doing something and are thinking about something completely different - for example when driving

home and taking the same route one have taken for years, people notice every now and then that they are driving completely "on autopilot". This habit creation is oftentimes very useful, if one had to think about every single partial task they are doing through the day - consciously tie their shoelaces, brush their teeth, insert a key into the lock and others, it would be very exhausting. However sometimes this behaviour is unwanted, counterproductive or even outright dangerous. Drinking, smoking, biting ones nails etc. These are not called bad habits by accident. The brain, although very sophisticated, cannot distinguish between a good and bad habit and when it gets the dopamine response, it remembers it very well(5). This is frustrating especially because when one tries to develop a healthy habit, it seems like their mind is sometimes working against them and refuses to perform the activity one would like to make into a habit like jogging or exercising. This is where motivation comes in - a properly motivated person has higher chances of managing to overcome the unwillingness to do some activity and accomplishing their goals. However the research behind habit making and changing ones behaviour shows, that motivation alone is almost never enough and if one tries to build new habit by forcing themselves into the activity over and over again, they rarely succeed.

2.5 How to build a habit

Very often one can read that it takes 21 days to build a habit and that if they force themselves into some activity for that period of time it will become an automated habit and they will be done. However that has been shown not to be true(6). The main problem is, that people are focused on the results, rather than the process. Take, for example, a person, who wants to be fit and healthy - they set a goal to lose 10kg of weight. They force themselves to run every other day and to eat mostly vegetables and manage to lose the weight they set out to lose. But what is next? They have achieved their goal and have no longer the motivation required to continue and very often slip to their old ways. The way to build a lasting habit is firstly to make it part of ones identity and then prove that it really is the case contrary to overdoing it at the start. Get better every day with small but steady steps. There are four main laws of building habits - Make it obvious, Make it attractive, Make it easy and Make it satisfying. This is according to the four step process of habit activation - One notices a cue which creates a craving for the activity, which motivates doing it which generates the reward, whatever it may be which finally satisfies the craving and reaffirms the habit and the connection between the cue and the reward.

Make it obvious - Habits work best when associated with cues(7). For example a cue might be a person coming home after work or seeing a painting on a wall in front of them when they sit in front of a computer or the smell of a room when they step into it. A person does not even need to be aware of these cues for them to work and create a craving. One of the best ways to start a new habit is to associate existing habit with a new one - which is

called habit stacking. But in order to do so, one must first be aware of their existing habits - this is best accomplished by a habit scorecard, which is a simple list of activities one does through the day and after a few days there can be visible patterns in ones behaviour. Other useful method for noticing existing habits is a pointing-and-calling method which means that a person should point to an object they are about to pickup or a place they are about to go and state the reason, why they are doing so, out loud.

Make it attractive - It was already said, that one of the main driving forces in habit making are the rewards(7). The rewards are the reason for a motivation to perform some activity as a habit is a dopamine driven feedback loop and it is the expectation of some reward that creates the dopamine spikes. In order to make a habit attractive, there is an easy method called temptation bundling - essentially one should do something they already enjoy after performing the desired activity they want to make a habit from. But it is also important for this rewarding activity to be aligned with the new identity one is forming and its values - more about this will be mentioned in a later section.

Make it easy - The human mind does what it does in order to conserve energy, it is a relic from times when food and water - the basic needs according to Maslow's(8) research and the pyramid of human needs, were not guaranteed and people spent most of their time trying to survive. This is however not an issue anymore for most of the people around the world. However even though basic needs are not the main issue for humanity anymore, in order to build a habit we need to conform to the ways our brain has developed over the millions of years of its evolution - it prefers doing easy things(7). A two minute rule is a simple tool addressing this issue - it states that rather than planning for a long time and thinking about how one is going to do something, it is better to just start doing it for a very short time every day. For example - exercising for two minutes will not help a person to get in shape, but it will help to build and standardize the habit and once that is done, it is easier to start for example running for 10 minutes every day instead of just two. Another helpful thing is to remove obstacles, rather than overcome them - to continue with the running example, if one wants to become a runner, they should prepare running clothes the day before, place them at a visible spot, plan the route and prepare a playlist to listen to.

Make it satisfying - It is best explained in one sentence - rewarding habits are repeated while punishing actions are avoided(7). It is known that the human mind prefers actions with immediate satisfaction but it is not good at recognising long term benefits of some actions - the key is to reward the behaviour or actions one wants to internalize "artificially" until the long term benefits and rewards present themselves. For example to continue with the running example - people know that if they keep running regularly for long enough they will get more fit, will be able to run longer distances etc. but it will be hard and will require a lot of willpower, however if the person decides, that after every jog they will read the news, take their phone and read what is new in their social networks feed or generally do something they enjoy

the running will get associated with the instant gratification of the reward activity and it will get easier.

■ 2.6 Plugging habits into motivation

It was presented in the motivation section, people deliver best performances when intrinsically motivated and intrinsic motivation is best inspired when the task is aligned with their identity, beliefs and when they feel confidence and competence. Later then in the section about habits the view of a respected personality in the field of habits was presented on why habits are important and also very useful in personal development(9) and how to best create and keep them. It was stated that in order to build a lasting habit, it should be aligned with ones personality and their beliefs and if it is not, one should shift their personality into someone, for whom these activities, they are trying to transform into habits, are aligned with their core values. After that various techniques and rules one should keep up with in order to build a habit were presented, which should not be seen in direct alignment with the principles of intrinsic motivation inspiration presented by the Self Determination Theory (1) but rather as "life hacks" to help a person to internalize the activity and it in the end becoming intrinsically motivated and aligned with their personality as well as transforming ones personality into one for whom this activity is core.

■ 2.7 Conclusion

While this chapter was focused on the theoretical part of the solution idea for a personal development application, in the following chapter there is a research of existing solutions of similar applications - mostly their features, philosophy on how to motivate people to stick with their desired habits and how to help them with it as well as how these applications guide their users along the way.

Chapter 3

A survey of relevant existing solutions

3.1 List of existing application and research methodology

When searching for other similar applications (10)(11)(12) to research and take inspiration from the following criteria for picking the applications for closer inspection were set:

1. *Should be habit or task/goal oriented*
2. *Should not be overly focused on meditation, quotes and/or book recommendations*
3. *Should be usable for free at least in a limited manner*
4. *Should have some interesting idea or functionality regarding motivating users or besides being just a tracker*

After setting these criteria the search for similar applications was conducted in the following order:

1. *Search in application stores, specifically App Store and Google Play*
2. *Blind search on Google for lists of applications*
3. *Select multiple applications from these lists which follow the second above listed criteria*
4. *Pick applications which follow the first and third criteria and are, ideally, present in multiple lists*
5. *Pick at least **three** and up to **seven** of those applications which have the most interesting idea behind them*

After all of these steps four applications remained which were chosen to explore in more detail.

3.2 Habitica

(13) Habitica is a multiplatform (iOS, Android, Web browser) habit tracking application. It differs between and tracks four main types of activities:

1. **Dailys** - The actions one does routinely on a daily basis such as brushing ones teeth, flossing, meditating and others. One can set a difficulty, add a note and tags to the action and also set days on which the action should be repeated as well as define a checklist of subtasks of which this activity consists of. And finally the application keeps a streak which means how many times in a row has one performed this activity.
2. **Todos** - One time actions that need to be done such as writing or reading an email, calling someone, going shopping etc.. One can create an event, set a time and date for it and once the time comes they get a notification reminding them of it. Once the task is done, one can swipe it off in the application and the task is marked as done and disappears. This type of activity can also be assigned a difficulty level, subtasks list can be defined and a due date upon which it should be done.
3. **Habits** - As the name of the application implies this is the most important element of the application. One can define the habit, add a note, set a difficulty which affects the rewards for sticking with the habit and a counter which can either only be incremented but can also be decremented when it makes sense. One can also add different tags to the habit which help with sorting and define a reset period so for example when the desired frequency of the habit is every two days, one sets reset period of two days.
4. **Challenges** - Habitica also has a social aspect which will be discussed later, but part of it are community challenges which are essentially pledges which users can participate in, they consist of some task which can be freely defined and one needs to submit proof of completion in a given time limit. After completing it the users are rewarded with the ingame currency.

Habitica introduces a very elaborate and well thought out system of gamification which means enhancing users motivation by implementing a game-like elements into the application. Each user has his avatar, for which they gain experience and ingame currency by completing different quests, challenges and sticking to their tasks and habits. One can buy new items with this ingame currency as well as for example health potion to replenish their characters health which is lost when for example one does not accomplish a task.

Positives

- Very well implemented gamification elements
- Good user interface and the ability to specify activities
- Social interaction functions and elements
- Multiplatformness

Negatives

- Limited functionality besides from the basic elements of habit trackers
- At first the gamification elements are nice, but do not go deep and are only superficial so the motivation quickly fades

3.3 Way of life

(14) Way of life is, in essence, very simple app for tracking habits and ones progress with their habits. Although it has passed the criteria of being usable without paying only technically (It can be used for free but only limited amount of times per day) given that it was found interesting and that it offers a one time payment method to use it without limitations instead of pay-per-month model, it was decided to include it in the list. Another reason for including it is that even though one must pay in order to use it fully, the app also guarantees data safety - meaning paying for the application grants enhanced privacy and the app does not sell ones data which was found to be a very good and honest business model. It offers the following features:

1. **Habit tracking** - A feature common to all habit trackers as the name would imply, but what was interesting specifically about Way of Life's take on habit tracking is that it is using so called chains instead of streaks, which will be described in more detail later.
2. **Notes** - Each time one completes or fails to complete a habit, they can add a note to the habits journal (each habit has its own journal) describing for example why they did not complete the activity this time or how did they feel about it which offers a more realistic approach to habit making, since nobody is perfect and this sort of personalised approach allows one not to feel bad for not completing the habit and get demotivated further.
3. **Chains** - Chains are Way of Life's way to approach streaks, one can set a desired streak length for a given habits so for example if they want to run every day, but also want to have some breathing room, they set a length of five days and if they manage to run five times in a row, their streak is not broken when they take a day off. These chains record a successful streaks as well as unsuccessful ones meaning how many times one failed to do the activity they set out to do which is a good indicator that one is slipping.
4. **Data collection** - Way of Life collects a lot of data about the user, which it, as it has been said already, does not sell or pass further, but

uses it to create detailed statistics about ones progress which I find very useful and this data can also be exported from the application in the form of .csv file, .xlsx file and other formats.

The most interesting thing about this application is the combination of data driven approach with its various statistics and graphs, the way the application approaches realistic expectations about ones ability to perform every single time and its sober attitude to data protection and subscription model.

Positives

- Chains feature
- Approach to data collection and business model
- Statistics
- Notes for habits

Negatives

- Although very well implemented and thought out, it is yet again just a habit tracker with no additional features
- Until recently it was not available for iOS platform

3.4 Habitify

(15) Habitify is yet another data driven habit application. While it means challenging the boundaries of the criteria, it was inspiring enough that it was decided to include it in the list. It offers a very simple design supported on all platforms - Windows, iOS, Android, smart watch, MacOS and it also has a in browser option. It offers habit tracking, where one can sort these habits based on a time of day they should be done at or by a different criteria set by the user. The most dominant features of Habitify are its statistics section and well implemented reminders as well as a habit journal which are very useful.

1. **Habits** - Each habit can be categorized by its label which can either be a time of day or a custom category and also each habit has its own journal and most importantly its own statistics section with streaks for this particular habit, completion rate and various indicators for possible future trends.
2. **Reminders** - Habitify works with an interesting reminder implementation and while it is not something new it is an important feature, since reminders and supportive messages are an important factor when building new habits.
3. **Statistics** - Besides the statistics for individual activities the application has a progress section tracking ones overall progress and trends.

The most interesting features worth noting are its statistics, simplicity and simple yet effective design. Oftentimes it seemed like some other applications tried to do too much at the same time and one would get eventually lost in all of their features and possibilites.

Positives

- Statistics section
- Simple design
- Journal

Negatives

- Its subscription model, although it offers a lifetime membership option
- Lack of additional features - even though its simplicity, it might be beneficial for example the option to differentiate between todo and a habit.

■ 3.5 Productive

(10) The Productive app has some part of everything from the other applications. The most interesting feature is that before launching the app the first time one is required to sign a statement that they will try their best to stick with their habits. It is a small thing but it helps to engage the user and make the process of habit forming more personal. It is usable without paying, even though there is a paid premium option, one can use it to a reasonable degree. The most important and interesting aspect of this application is the intro questionnaire which determines ones experience in the process of habit forming and based on the answers the app guides the user and helps them them stick with their habits. It offers the following features:

1. **Habits** - The habits feature is similar to the other applications, one can choose a category from predefined set which is quite vast or create their own category, habits can be sorted by daytime and place and can be assigned a timer to measure time set for the particular activity.
2. **Challenges** - Challenges are activities with set time in the week one is supposed to perform them, for example the challenge might be to exercise on Monday and Wednesday and one must sign a pledge to complete the challenge in order to participate.
3. **Articles** - There is a section dedicated to various related articles one may read and find motivation or inspiration in.
4. **Statistics** - Similarly to other applications Productive too offers a statistics section recording ones ups and downs and their performance.

The most interesting feature in this application is the guidance one receives after filling in the initial questionnaire. It offers a sort of relatedness and may add to users motivation and help them stick with their habits. The unfortunate factor of this application is, that the entire statistics section is hidden for non-paying users.

Positives

- Simple application design
- Pledges and guidance along the way
- Social engagement with challenges

Negatives

- Statistics section hidden behind paywall
- Severe lack of additional functions, even though the basic functionality is very well thought out.

3.6 Summary and conclusion

After reviewing these other applications here are the highlights of some key concepts I found very useful and worth taking inspiration from.

1. *Simplicity of design*

Simplicity in both feature design and visual appearance of the app is key to making a good application which users will return to. Initially it was thought that the application should have more features, but after seeing and trying some of these applications the focus has shifted towards simplicity - fewer functions but very well thought out and implemented.

2. *Journaling option and note taking*

Journaling is important feature for developing a personal affection for the activity and reaffirming it as aligned with ones beliefs.

3. *Statistics and progress tracking*

If well implemented and visually appealing, progress tracking should be one of the core features of such application. It can offer insight into ones trends and tendencies and even if not that, it is very appealing for people to see their progress and is one of the most important motivators.

Application	Positives	Negatives
Habitica	Gamification, UI, Social features 2	Otherwise limited functionality, Short spanned gamification attractiveness
Way of life	Approach to data, Statistical features, Note taking	No additional features
Habitify	Statistical features, Design simplicity	Subscription model, Lack of additional features
Productive	Design simplicity, User guidance, Social features	Statistical features only for premium users, Lack of additional features

Table 3.1: Existing solutions comparison

The above table is a summary of the above mentioned positives and negatives of the individual examined applications.

In conclusion all of the applications above had their pros and cons, the negatives being mostly the business model and also the lack of tools for the user to really align their personality with their process and make it part of them. But on the other hand they offered an interesting insight and helped to realise some of the key concepts, as mentioned above, on which the application might be built.

Chapter 4

Requirements and use cases

4.1 General concepts

In this chapter the functional and non-functional requirements for the application will be presented and in later chapters there will be the presentation of the technological stack upon which the application should be built. But first there is a presentation of the general concepts which are the philosophy behind the application. As was presented in the previous chapters, the key concepts for building an application which helps people stick with and build their habits are first and foremost to make them aligned with ones personality. To help with this the take of an author of the book Atomic Habits (7) was presented which condenses down to four main rules - make it easy, make it attractive, make it satisfying and make it obvious. These rules indirectly correspond with the research of Self Determination Theory (16) and intrinsic and extrinsic motivation and it will be presented how the concepts, the application will be built on, implement these rules.

The first concept is for the application to have simple and effective visual design. When researching other similar applications it was observed that the most liked applications had a simple and straight forward visual design. This can also be seen for example with the products of Apple. Should users be using the application long term, there must be as few obstacles for them as possible and this includes the design itself. It is believed that a lot of colors and shapes is a thing of the past and the majority of products today are leaning towards minimalism as well. It makes the application organised and enjoyable to use, both of which align with the rules of making the process easy and attractive.

Second concept is feature minimalism. Too many options are not necessarily for the best. The application should have clearly defined functionalities which are easy to understand and use and implement all of the functional requirements, but do not allow for excessive customization. From experience, customization of features and the option to over-specify them are attractive at first, but after a while create a chore, when one must spend extra time on specifying the details instead of keeping their focus on what is important, in this case - building habits, making life more organised and spending less time on management, rather than more. In summary - The users should have

clearly defined and well thought out options to choose from which provide for all of their needs, but do not allow for an extra dwelling on the details, because the users come to the application for help and guidance and it should provide precisely that.

Data orientation and strong statistics section - people like to see their progress, although it is an external motivator, it is true and it motivates them to continue when they see that they have already something behind them. Since many of activities which are beneficial for people, such as exercising, better time management, etc. do not offer an immediate reward but rather a delayed one, in this case a better physical condition or more free time and less stress, seeing their progress can and often is the motivator to keep them going until the actual rewards present themselves. This has been specifically argued in the book *Atomic Habits* (7) - not only that keeping track of ones progress is a good motivator, but also the concept of artificial intermediate reward until the activity itself starts paying dividends. Another positive side effect of good record keeping of ones progress is that it can spot negative trends rising before the user does in real life and can serve as a reminder not to go down the rabbit hole.

In summary simplicity and efficiency of the visual design, feature minimalism and strong emphasis on statistics are the key concepts the application should be built on because they are not only good practice but align with the rules of making habits and human psychology. In the next section the functional and non-functional requirements will be presented, which implement the functionality of the application and adhere to these concepts as well as their individual importance and estimated difficulty.

4.2 General functionality draft

In this section the functionalities the app will have based on the research of motivation, habits, habit making and other similar applications will be introduced.

4.2.1 User

User entity is the founding stone of many applications and represents the user itself in the application.

1. User is created via registration
2. Registration is done via an email bound to this user
3. User has an avatar
4. User can input general information about them such as age, weight, height, gender and others

■ 4.2.2 Habit

Habit is an event which implements the core functionality of habit as was explained in earlier section.

1. Habit has name
2. Habit has at least one category or more
3. Habit has a frequency of how often should be performed
4. Habit has specified length of how long does it take to perform it
5. Habit has optional description and a note
6. Habit has its own small statistics section

■ 4.2.3 ToDo

ToDo is a one time event, which represents a reminder for the user to do something such as write and email for example.

1. ToDo has a name
2. ToDo has a category
3. ToDo has an importance level
4. ToDo has a due date

■ 4.2.4 Calendar

A calendar which may be connected with the users other calendars and contains the planned events like Habits or ToDo's

1. Calendar contains the events made in the application
2. Calendar can be connected with other calendars

■ 4.2.5 Statistics section

A section of its own (compared to the statistics section of each habit) with overall statistics, trends and various filters.

1. An individual habit trends
2. Overall habit trends
3. Filters for different activities statistics
4. Predictions

■ 4.3 Functional requirements

■ 4.3.1 Landing page

- When a potential new user finds the application, they must find a pleasant landing page which will shortly but effectively introduce them into what the application can do and how can it help them and ideally gets them to sign up. Must contain a link to either log in or sign up for the application.
- Expected difficulty: Low
- Priority: High

■ 4.3.2 Registration

- An unregistered user must have an option to register for the application
- Expected difficulty: Moderate
- Priority: Very high

■ 4.3.3 Login

- An already registered user must have a way to log in to the application
- Expected difficulty: Moderate
- Priority: Very high

■ 4.3.4 User profile

- A user needs a way to manage their profile, information and data about them.
- Expected difficulty: High
- Priority: Medium

■ 4.3.5 User dashboard

- When logged in first thing a user sees is an overview of their todos and habits
- Expected difficulty: High
- Priority: High

■ 4.3.6 Habit

- User can create a habit event, specify the details, see related statistics and recommendations. The habit details are defined by their category.
- Expected difficulty: High
- Priority: Very high

■ 4.3.7 Habit list

- User can browse their habits on a side panel of the habits tab.
- Expected difficulty: Moderate
- Priority: Moderate

■ 4.3.8 Todos

- User can create a todo event, select category, specify details and mark it as done to make it disappear.
- Expected difficulty: High
- Priority: High

■ 4.3.9 Todo list

- User can browse through their todos and filter based on criteria.
- Expected difficulty: Moderate
- Priority: Moderate

■ 4.3.10 Calendar

- User has a personal calendar.
- Expected difficulty: High
- Priority: Moderate

■ 4.3.11 Calendar events

- User can add their planned Todos and Habits to their calendar along with other custom events
- Expected difficulty: Moderate
- Priority: Moderate

■ 4.3.12 Statistics section

- User can see various statistics from their current progress, trends of their behaviour regarding Habits and Todos and projection of future progress
- Expected difficulty: Very high
- Priority: High

■ 4.4 Use cases

In this section the individual ways how one can use the applications, features and what parts of the application serve these use cases will be introduced.

■ 4.4.1 User wants to register

- An unregistered user wants to sign up for the application, so they arrive at the landing page, which sends them to the sign up form.
- Involved entities: Unregistered user, Landing page
- Covered by: Landing page

■ 4.4.2 Registered User wants to log in

- An already registered user wants to enter the application, so they arrive at the landing page, which sends them to the login form.
- Involved entities: Registered user, Landing page
- Covered by: Landing page

■ 4.4.3 Registered user wants to log out

- A registered user wants to exit the application, so they click the log out button on the toolbar and the applications signs them out
- Involved entities: Registered user, log out button
- Covered by: Toolbar, Log out button

■ 4.4.4 Logged in user wants to change their personal information

- A logged in user wants to change some of the information about them, so they click on his profile icon which sends them to the user profile page
- Involved entities: Logged in user, Toolbar, User profile
- Covered by: Toolbar, User profile

■ 4.4.5 Logged in user wants to delete their profile

- A registered,logged in user wants to delete their profile, so they click on their profile icon which sends them to the user profile page
- Involved entities: Logged in user, Toolbar, User profile
- Covered by: Toolbar, User profile

■ 4.4.6 Logged in user wants to see an overview of their activities

- A logged in user wants to see their current and future events, most important information, general trends or missed events, so they clicks onto the User dashboard icon on the navigation bar which sends them to the User dashboard page
- Involved entities: Logged in user, Navigation bar, User dashboard
- Covered by: Navigation bar, User dashboard

■ 4.4.7 Logged in user wants to see some upcoming event displayed on their dashboard

- Logged in user wants to see details about an upcoming Todo or Habit event, so they click onto the event tile on their dashboard which sends them to the events tab in either Todo section or Habit section
- Involved entities: Logged in user, User dashboard, either Habit section or Todo section
- Covered by: User dashboard, either Habit section or Todo section

■ 4.4.8 Logged in user wants to see specific habit/Todo

- A logged in user wants to see a specific activity and related details, so they click the appropriate tab on navigation bar and then find the specific activity via filter column.
- Involved entities: Logged in user, Navigation bar, Habit/Todo section, Filter column
- Covered by: Navigation bar, Filter column, Habit/Todo section

■ 4.4.9 Logged in user wants to create an activity

- A logged in user wants to create a new Habit or Todo, so they get via the navigation bar to the appropriate module where they click on the add button on the toolbar and fill in the details.

- Involved entities: Logged in user, Habit/ToDo module, Navigation bar, Tool bar
 - Covered by: Habit/ToDo section, Toolbar
- **4.4.10 Logged in user wants to see their statistics and analysis of his current performance and behaviour**
- A logged in user wants to see their analyzed data, so they get to the statistics module via navigation bar and select the section they want to see.
 - Involved entities: Logged in user, Navigation bar, Statistics section
 - Covered by: Navigation bar, Statistics section
- **4.4.11 Logged in user wants to see their schedule**
- A logged in user wants to see their calendar, so they click on the icon on navigation bar and get to calendar.
 - Involved entities: Logged in user, Navigation bar, Calendar
 - Covered by: Navigation bar, Calendar
- **4.4.12 Logged in user wants to add a custom event to their calendar**
- A logged in user wants to add some event other than Todo or Habit to their calendar, so they click on the add event button on the toolbar and fill in the details.
 - Involved entities: Logged in user, Calendar, Toolbar
 - Covered by: Calendar, Toolbar

Chapter 5

Technological stack and architecture proposal

In this chapter the results of the research will be introduced regarding the possible tools such as programming languages, programming frameworks, database systems etc. that might be suitable for an application such as this and then discuss the choice of individual tools for each area (Frontend, Backend, Database, ...) upon which the application will be built.

5.1 Basic software application architectural patterns

Software architectural patterns are ways to split the application into parts each of which has a specific purpose in the application and does specifically only one thing, ie. one part deals with data manipulation, other handles requests from Frontend and responses from Backend and another one handles User interface and rendering of data.

5.1.1 Model View Controller (MVC)

(17) A Model View Controller is the perhaps most used architectural pattern in applications. The three main parts Model, View and controller each handle specific tasks for the application.

1. Model - Is the Backend of the application, manipulates and stores data and returns manipulated data according to the user to the Controller.
2. View - Is the Frontend of the application, defines how the data the user requested will be organised and look like visually.
3. Controller - Is the middleman between the Frontend and the Backend. It handles user requests and translates these requests to the Model and then presents the manipulated data on the Frontend.

5.1.2 Model View Presenter(MVP)

(18) It is in a way an alternative to the MVC architecture. The difference is that the View has more "power" and is not static. The View gets user input,

creates its presenter and then interacts with the Model via an interface.

1. Model - Same role as in MVC, has data and manipulates them according to requests from above and returns manipulated data.
2. View - Contains business logic, on input/user requests creates presenter to interact with Model.
3. Presenter - Handles requests from the View which uses it to communicate with the Model, prepares the data obtained from the Model and returns them to the View for presentation

5.2 Server side Web application frameworks

As it does not seem useful trying to build the application blocks separately since there are many frameworks which handle all of these areas at once and are the most used instead research these frameworks will be researched and the one which seems most interesting and best made will be chosen.

5.2.1 Django

(19) Django is a web application framework written in python. It covers all of the layers of the application architecture, gives emphasis on speed and security and has a lot of things already implemented, so one can focus on building an application from the blocks already prepared in Django, instead of writing everything from scratch. Its advantages are firstly quick development partly thanks to the python language in which Django is written, security, versatility and very detailed documentation as well as backwards compatibility. The basic building blocks of which Django consists are:

1. Urls - When user types a url in their browser, the request first arrives at the urls. It is a place in the application where patterns are defined according to individual modules of the application (for example "www.myapplication.cz/habits") and each request is compared to these patterns until a match is found in which case the appropriate view is called. Otherwise an error is returned.
2. Views - View is the center of the application, after a request is routed to the appropriate View, the view is given the parameters and defines what should be done, in the above example of "www.myapplication.com/habits" it queries data from the Habit model, for the logged in user, transforms them appropriately and does other related tasks and then sends the queried and formatted data to the Template.
3. Templates - A Template defines what the user sees upon loading the site they requested. After the View is done, it sends data to the template which renders them appropriately along with other elements of the web page and lets the user see them and interact with them. The Template

uses Django Template Language which allows creation of dynamic website instead of just static representation, but can also be amended with javascript or typescript or other frontend languages.

4. Model - A Model is a representation of an object in the application or more specifically representation of a database table. Using Object Relational Mapping it allows for data manipulation via python code and generally for handling data in the application logic.
5. Django ORM - Object relational mapping is a way for comfortable accessing a database from a programming language. Each database table has a corresponding model which has fields which correspond to the db table columns. On instances of these objects one can perform various actions which are translated into an SQL query in the end.

■ 5.2.2 Spring Boot

(20) Spring Boot is a web application framework based on the programming language Java. Its main advantages are scalability, security, easy testing, the option for multithreading and automatic configuration. Compared to Django it is more suitable for large projects, but for medium to small projects it seems too heavy. Also as it is based on Java it is more complicated to get started and write the application from scratch compared to python. On the other hand it is much more suitable for large projects with large teams as it is easier to enforce architecture and coding style in Spring compared to Django. Spring Boot does not have, as opposed to Django, a hard given structure and the project structure is defined by developer.

■ 5.3 Database

■ 5.3.1 MySQL vs PostgreSQL

(21) Initially it was intended to dedicate a subsection to both of these individually but with more research it became clearer that both of these database systems have very much in common and for the purposes of this application there is very little that distinguishes them from each other. Both PostgreSQL and MySQL are relational database systems which excel in security, speed, scalability, supported programming languages and both are open source. The main difference is, that PostgreSQL is more sophisticated and allows for more data types and is more extensible, which reflects in slightly more effort it takes to work with it. PostgreSQL also supports multithreading and MySQL does not. On the other hand it is easier to work with and deploy MySQL database.

■ 5.4 Frontend frameworks

■ 5.4.1 Vue JS

(22) Vue is a frontend framework based on Javascript. It has something from both React JS and Angular (also Javascript frontend frameworks which will be described later). It has a fast learning curve, meaning it is easy to understand and use. Also Vue uses a modular approach and is component-based meaning one creates a component such as for example a button and then it is possible to include the button on multiple places in the application. The page created in Vue is an HTML template with mounted Vue functions on elements and included components.(23)

Advantages:

- Component based approach
- Uses virtual DOM
- Lightweightness
- Customization

Disadvantages

- Extensive flexibility
- Low scalability
- Small community and limited support

■ 5.4.2 React JS

(24) React JS is another Javascript based frontend framework. It also uses a modular, component based approach and templating. Similar to Vue, React has a swift learning curve. Compared to Vue, React uses a RJSX Javascript language extension instead of single file components, which essentially means that the HTML code is integrated into the Javascript code.

Advantages:

- Swift learning curve
- RJSX extension
- Scalability
- Easy SEO optimization

Disadvantages:

- Poor documentation
- Really fast development of the library - meaning constant changes

■ 5.5 CSS Frameworks

■ 5.5.1 Bulma.io

(25)(26) Bulma is one many css frameworks, is opensource and its aim is for its users to code as little as possible. It has lot of built ins and components and can be imported by parts. It is also a strictly CSS framework, meaning that compared to some other frameworks it does not rely on Javascript or Ruby. It also has relatively good default state, meaning that if one wants to just use its classes and not alter the css, it still looks nice visually.

Advantages

- Very easy implementation
- Easily readable classes - and easy understanding of what they do
- Quick learning curve
- Nice default visual design

Disadvantages

- Hard customization of classes
- Very distinctive visual design which is hard to alter
- Counter-intuitive documentation

■ 5.5.2 Bootstrap

(25)(26)(27) Bootstrap is one of the most - if not the most - popular CSS framework today. It is mobile first and works with separation of the page into columns and rows. The new Bootstrap 4 is building on the flexbox features of CSS. While it also is heavily opinionated, meaning the framework has its way of doing things it deems right, its not as hard to style some elements differently and still keep unified design, compared to Bulma.

Advantages

- Supported by large company - Twitter
- Very quick prototyping
- Very large collection of components

Disadvantages

- Less understandable naming of classes
- Rendering can sometimes lead to a lot of HTML code
- Requires more work for the site to look unique

5.6 Technologies selected for the implementation

5.6.1 Web application framework

As for the backend framework, it was decided to use Django framework, for the following reasons:

1. Is based on python as opposed to Java which seems to allow for clearer code and more rapid development.
2. Is more lightweight, which is more fitting for this project as the application is to be relatively lightweight as opposed to for example an internet banking application.
3. Is easier to understand and requires less initial effort to get started.
4. As was already mentioned, Django is based on python language which brings another great advantage and that is the ability to easily integrate and use its vast libraries and frameworks for statistics and AI which are intended to be used in the statistics section.
5. The author already has moderately vast knowledge and skill with programming in python language and specifically creating and maintaining Django-based application and would like to deepen their knowledge of this framework.

Django security

Security is one of the most important factors to consider when choosing the frameworks for a web application which will be handling user data. One of the pillars of Django framework and its pride is its security. Django provides security measures and protections against the following types of attacks and many more(28):

1. Cross-site scripting
2. SQL injections
3. Cross site request forgery
4. Clickjacking

Of course the Django framework, same as any other framework, is not immune to attacks, but it has a good baseline of security after which it us on the developer to follow the safe practices in web development. One of these very important practices is for the frontend application to communicate with the backend via HTTPS protocol to assure it is a secure communication.

■ 5.6.2 Database

PostgreSQL was selected as the database system for this application, for the following reasons:

1. As was mentioned before - from the point of view of this project and its requirements there are many similarities between these two systems. In the end PostgreSQL was selected solely for its slightly better extensibility and the possibility to store more datatypes.

■ 5.6.3 Frontend framework

For the frontend framework, it was decided to use the React JS framework, should the need arise for such framework, as Django already has functionality of a frontend framework to some degree, for the following reasons:

1. It is easier to learn
2. It is widely used today
3. The author would like to extend their technological stack with this framework.

However if there is no need for larger Frontend framework a Javascript XHR framework will be used, as it is more lightweight and is better suited if there will be only a few requests which will have to be sent from the frontend.

■ 5.6.4 CSS Framework

And finally for the CSS framework, it was decided to go with Bootstrap, because:

1. It is widely used
2. The author has worked with both Bulma and Bootstrap and in their opinion, Bulma is very lightweight, easy to implement and has a nice default, however it was found very hard to alter the style of the elements, because then one has to change the appearance of all of the elements, otherwise the individual ones stick out.
3. Author would like to extend their knowledge of this framework as well.

■ 5.6.5 Database model draft

The logical database model will consist of the following tables:

- Users
- Habits
- Todos
- HabitPlan
- HabitSession

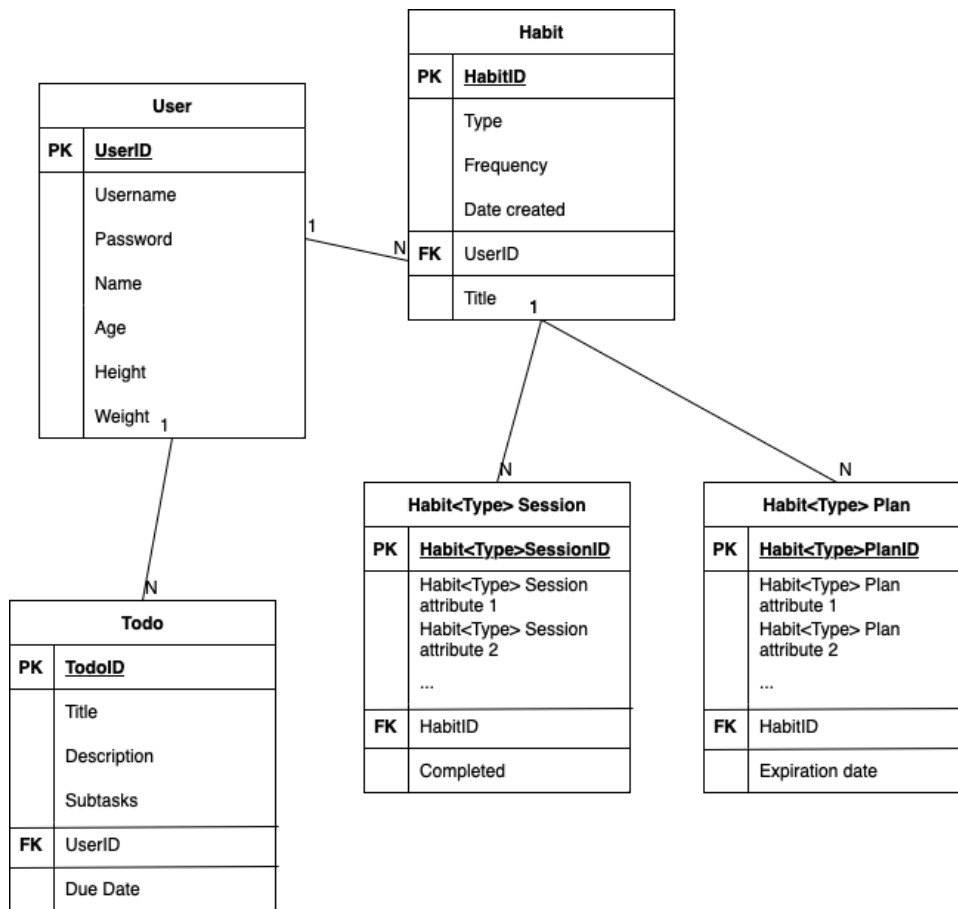
The actual database will have more tables, where for each Habit type there will be separate Habit<Type>Session and Habit<Type>Plan tables. This is due to the fact that for each Habit type a different type of information will be required to be stored - for example for an exercise habit there have to be columns such as calories burned, kilometers ran, duration, average heart rate etc., on the other hand for a learning habit there have to be columns such as subject, pages read, duration and others. The alternative to this is having sparse tables with many columns where for each row would have most of the column values empty and this way would be very confusing, prone to mistakes, vulnerable from the database integrity point of view and would make the work with Django significantly harder.

■ Users table

The users table will contain information about the user such as password, name, username, age, height, weight and others. There will be only one row in the Users table for each user and it will be in a one to many (1:N) relationship with the Habit table and the Todos table.

■ Habits table

The Habits table will contain general information about each Habit for each user such as date created, title, frequency and other information which must be kept for all habits regardless of its type. The Habits table will be in a one to many (1:N) with the users table (from the point of view of the Users table - meaning one user may have many habits) and then it will also be in a one to many (1:N) relationship with one from each of the Habit<Type>Session and Habit<Type>Session tables - meaning each Habit may only have one type but it has to have one type and there may be many HabitPlans for each Habit (since the HabitPlan may change over time, but it should be stored historically) and there will be many HabitSession records for each Habit since the HabitSession record represents one repetition of the Habit.



Note: Due to the spacial complexity of a fully-fledged ER diagram, it was elected to simplify it (it would not fit in the document)

Figure 5.1: Database Schema

Habit<Type>Plan table

There will be one Habit<Type>Plan table for each Habit type and it will store information about the goal the users set for themselves for the particular Habit - for example if the user wishes to run 10km in 1 hour every session and burn 400 kcal then this will be stored in their plan. They can alter the plan whenever they want, however the past plans will be stored so it will be possible to calculate precise statistics such as adherence to their plan for some period of time. The Habit<Type>Plan tables will be in a one to many relationship with the Habits table (from the point of view of the Habits table) meaning each Habit may have many plans connected with it.

Habit<Type>Session table

There will also be one Habit<Type>Session table for each Habit type and it will contain information about the users actual performance when they do the activity they wish to make a habit of - for example if the user runs on a

Tuesday and Friday there will be two `HabitExerciseSession` records and each record will contain the information about their performance in the session (to continue with the running example, each record will contain the number of kilometers the user ran, the number of calories they burned, the duration of the session, date of the session).

■ Todos table

On the other hand there will be only one table for `Todos` as they are defined only by their title, description, due date, importance and subtasks which are all user defined (meaning they are text fields and the user can write whatever they want to store as a `Todo` into them). The `Todos` table will be in a one to many (1:N) relationship with the `Users` table (from the point of view of the `Users` table) meaning one user may have many `Todos`.

Chapter 6

Implementation

In this chapter the actual implementation will be presented as the result of the previous domain analysis and a documentation of the process as well as a presentation of the difficulties which were faced and illustrative figures.

■ 6.0.1 Application modules

As was proposed in the fourth chapter the application consists of 5 main modules - Habit module, Todo module, Dashboard module, Calendar & Users profile where Habits, Todos and Users profile are the core modules of the application and the most important for the main functionality more emphasis was put on them.

■ 6.0.2 Environment preparation

In order to be able to focus solely on development and make the work easier a virtual environment was prepared and also the frameworks used to build the application with were installed.

■ Virtual environment

First of all in order to be able to have control of all installed python packages used in the project a python virtual environment was created using pyenv(29). Python virtual environment is a closed and encapsulated python environment which must be activated in order for a person to be able to work with it and it is isolated from the default system python version. It is very useful when working on larger project such as this application since it is easy to keep track of which python packages are installed, in what version and it makes it easier to be sure that there are no conflicts.(30)

■ Installing Bootstrap

There are several ways to install bootstrap, the easiest way was chosen due the fact that this application is running on Django was and so the python package django-bootstrap-v5 was installed.

■ Git

To make development easier, avoid mistakes and split the development into shorter, achievable and clear steps a git repository was created, using the faculty gitlab server.

■ 6.0.3 Helper structures

In order to keep the code itself clean and clear helper structures were created for all modules. A helper is a class related to a module in the application which contains various utility (but not only utility) functions which are used often and are as general as possible and so can be used in many places in the Backend of the application (mostly in API). This has helped to keep the code directly in the endpoint views to a minimal amount and also since the actual code is stored in one place (in a function of a helper) which is then called from multiple API endpoints a lot of time was saved when there had to be changes to the code since the code had to be changed in only one place instead of rewriting code in each view separately.

```

247 class HabitOverviewView(View):
248
249     new *
250     def __init__(self):
251         super().__init__()
252         self.helper = HabitsOverviewHelper()
253
254     new *
255     def get(self, request, *args, **kwargs):
256         print(request.user)
257         if not request.user.is_authenticated:
258             return redirect('login')
259
260         habit_slider_context = self.helper.get_data_for_habit_slider(request.user)
261         popup_forms = self.helper.get_forms_for_habit_submission()
262         plan_edit_forms = self.helper.get_data_for_habit_plan_edit(request.user)
263         habit_creation_form = self.helper.get_forms_for_habit_creation()
264         hero_habit_data = self.helper.get_hero_display_data(request.user)
265         ret = {'cards': habit_slider_context,
266              'habit_category_enum': HabitCategoryEnum.__members__,
267              'popup_forms': popup_forms,
268              'plan_edit_data': plan_edit_forms,
269              'create_forms': habit_creation_form,
270              'hero_data': hero_habit_data}
271         return render(request=request, template_name="habits/habits.html", context={'habit_slider_context': ret})

```

Figure 6.1: Example of a helper structure usage in an API endpoint

```

25 class HabitsOverviewHelper:
26     model = Habit
27
28     def __init__(self):...
29
30
31     def get_by_id(self, habit_id: int) -> Optional[Habit]:...
32
33
34
35
36
37     def get_all_by_user_ordered(self, user: User) -> QuerySet[Habit]:...
38
39
40
41     @staticmethod
42     def get_value_from_codebook(codebook_category: int, codebook_value: int) -> Optional[str]:...
43
44
45
46     @staticmethod
47     def get_last_completed_planned_session_by_type(habit: Habit):...
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64     @staticmethod
65     def get_last_session(habit: Habit):...
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82     @staticmethod
83     def get_next_habit_session_date(habit: Habit):...
84
85
86
87
88
89
90
91
92
93
94
95     def get_last_session_dict(self, habit: Habit):...

```

Figure 6.2: Example of a Helper structure implementation

As can be seen in the figures above a helper contains functions which may be used in many other places (general methods like `get_by_id` or `get_all`) and also help with code clarity.

6.0.4 Codebooks

Another utility structure is a Codebook. Since there was a lot of work with categorical values such as a frequency of habit repetition (Every week, every month, ...) or habit categories, learning subjects etc. there was a need for many Enum classes. Since storing literal string values of categorical variables in the database did not seem like a good practice, an Enum was used and when user for example enters 'Learning' as a category of their new habit the Integer value assigned to that value is saved instead. But when loading data from database to display them to the user there would have been a need for a lot of **if else** branching to display the right value which would hurt the code clarity a lot. In order to prevent this two database tables were created - **Codebook-Category** and **Codebook**. CodebookCategory contains the integer values assigned to each individual Enum and Codebook table contains a foreign key column referencing the CodebookCategory table which indicates to which Enum class this record belongs to and also a 'name' column containing the literal string value of the variable and finally 'value' column which corresponds to the value in the Enum class representing the 'name' column. It is then easy for example when querying a 'frequency' column value from a Habit table with the value one it is possible to easily query from Codebook table where `category_value` equals to `CodebookCategoryEnum.HabitFrequency.value` and `value` equals to the Habit 'frequency' column value. This has another added benefit - it allows for easy translation of these categorical variables since it will be possible to just extend the Codebook table by a column 'name_eng' which will contain the english translation of the name of the string value.

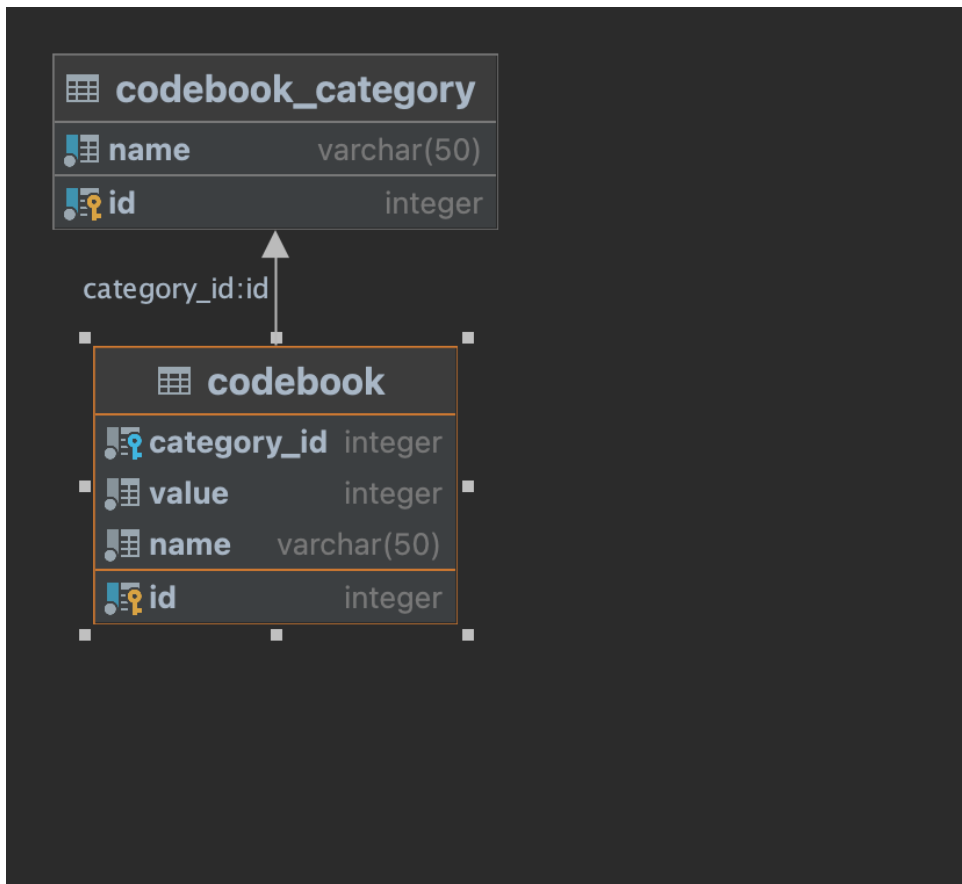


Figure 6.3: Codebook database tables

6.1 Backend development & API

In this section the documentation of the implementation process is presented, it will be shown how the most important features of the application are implemented and the solution will be discussed. The solution will be presented in blocks corresponding to the logical modules of the application (such as Habits, ToDo's,...).

6.1.1 Login & Registration

In order to be able to register a user and create their profile in the application it was necessary to create a landing page where every potential user or a logged out user will be redirected every time they visit the application so they can either sign up for the application or log in if they already have a profile. At the same time the landing page needed to be a pleasant welcome for the new users in order to attract them to sign up. Inspiration was drawn from (31)

Login

The already prepared Django functionality for authenticating users and creating users profiles was used and the creation of record in the table UsersProfiles with more information was added. Firstly there is a validation,

```

34 def login_request(request):
35     if request.method == "POST":
36         form = AuthenticationForm(data=request.POST)
37         if form.is_valid():
38             username = form.cleaned_data['username']
39             password = form.cleaned_data['password']
40             user = authenticate(username=username,
41                               password=password)
42
43             if user is None:
44                 messages.error(request, "Authentication failed")
45                 return render(request=request,
46                               template_name="login/login.html",
47                               context={'login_form': AuthenticationForm(),
48                                       'fail': True})
49
50             login(request, user)
51             users_profile = UserProfile.objects.filter(auth_user_id=user.id).first()
52             users_profile.last_login = datetime.datetime.now().astimezone(pytz.UTC)
53             users_profile.save()
54             return redirect("habit_overview")
55         print(form.errors)
56         return render(request=request,
57                       template_name="login/login.html",
58                       context={'login_form': AuthenticationForm()})
59     form = AuthenticationForm()
60     return render(request=request, template_name="login/login.html", context={'login_form': form})

```

Figure 6.4: Login API endpoint

checking that the request is of POST method and that the form sent with it is valid. After that it is attempted to authenticate the user using the **authenticate** function and if that authentication is successful, then user will be redirected to their dashboard. If any error occurs, an error message will be passed to the user and they will be asked to try again.

Registration

Again it was made use of the fact that Django already provides a well adjusted functionality for handling of logging in, creating users and handling users rights as well as the submitted data validation such as making sure that the email address is valid and that the password entered is sufficiently complex. After that there was added the creation of a record in the UsersProfiles table. Firstly there is a check of whether the request is of the right method and whether the Django registration form is valid. If that is the case then two entities are created - User and UserProfile. The user will then be logged in and redirected to the Dashboard module so they can start using the application. And if the form is not valid an error message is displayed to the user and they are asked to try again.

```

13 def register_request(request):
14     if request.method == "POST":
15         form = UserRegistrationForm(data=request.POST)
16         if form.is_valid():
17             messages.success(request, "Registration successful.")
18             new_user = form.save()
19             users_profile = UserProfile.objects.create(username=form.cleaned_data['username'],
20                                                         auth_user=new_user,
21                                                         email=form.cleaned_data['email'],
22                                                         last_login=datetime.datetime.now().astimezone(pytz.UTC),
23                                                         is_premium=False)
24             users_profile.save()
25             new_user = authenticate(username=form.cleaned_data['username'],
26                                     password=form.cleaned_data['password1'])
27             login(request, new_user)
28             return redirect("habit_overview")
29             messages.error(request, f"Unsuccessful registration. Invalid information. {form.errors}")
30         form = UserRegistrationForm()
31     return render(request=request, template_name="login/registration.html", context={"register_form": form})

```

Figure 6.5: API endpoint for registering

6.1.2 Habits

The Habits page is the most important module in the whole application and everything revolves around it so it was carefully planned out what the page should look like and what functionalities it must have.

Habit creation

The most basic functionality is the creation of a habit - It was necessary to create a Django form for submitting the information by the user, while keeping in mind that there are multiple habit types with different types of information that had to be saved. As a consequence a parent form was created, containing the fields which are common to all habit types inheriting from Django Form class and after that multiple forms which inherited from this parent form and have extra fields, specific for their habit type were created as well. There is a common API endpoint used for loading of all the data for the page for rendering - data for creation of habit, editing a habit plan, adding a habit session and others. For the other direction - submitting of the data by user and saving it into the database, there was a specific endpoint created for creation of a habit. Firstly it is needed to validate the submitted data. Along with the submitted form data the endpoint gets a URL parameter **habit_category** which indicates what category should the newly created habit be of. After creating the Habit object itself it is also required to create a HabitPlan object related to this habit and also plan the next session of the Habit based on the field **days_to_repeat** which indicates on what days does the user want to repeat the activity (it is a list of integers, valued from 0 to 6, indicating the weekday). Since it is needed to create multiple objects which are related to one another an atomic transaction⁽³²⁾ was used so that if an error occurs in the middle of creating these objects, none of them would be saved and the database integrity remains intact.

```

247 class HabitOverviewView(View):
248
249     new *
250     def __init__(self):
251         super().__init__()
252         self.helper = HabitsOverviewHelper()
253
254     new *
255     def get(self, request, *args, **kwargs):
256         print(request.user)
257         if not request.user.is_authenticated:
258             return redirect('login')
259
260         habit_slider_context = self.helper.get_data_for_habit_slider(request.user)
261         popup_forms = self.helper.get_forms_for_habit_submission()
262         plan_edit_forms = self.helper.get_data_for_habit_plan_edit(request.user)
263         habit_creation_form = self.helper.get_forms_for_habit_creation()
264         hero_habit_data = self.helper.get_hero_display_data(request.user)
265         ret = {'cards': habit_slider_context,
266              'habit_category_enum': HabitCategoryEnum.__members__,
267              'popup_forms': popup_forms,
268              'plan_edit_data': plan_edit_forms,
269              'create_forms': habit_creation_form,
270              'hero_data': hero_habit_data}
271         return render(request=request, template_name="habits/habits.html", context={'habit_slider_context': ret})

```

Figure 6.6: The central endpoint for loading of the Habit page

Editing of a Habit/Habit plan

The habit entity is, from users point of view, represented by a Habit plan (it contains the important information for the user) - it represents users goals for this particular habit and so it must be editable if the user wants to adjust their goals. Once again Django forms were used where a parent form was created, containing the fields which are common for all habits, this time using inheritance from the Django ModelForm class(33) which is a very useful class - one specifies the model, on which this form is based on in the constructor and the model then automatically contains all the fields of its model. And then specific forms for each habit type, inheriting from this parent form. As was mentioned before, the data for the initial display of the form are sent at once from the central endpoint but in this case the form has to be pre-filled its current values. Another specific thing about editing the Habit plan is that the rows are not overwritten but a new row is created and the old one is invalidated (expiration date column value is set which means it is no longer valid and will be used only for statistics computation). This allows to know what goals the user had in what periods of time and makes it possible to compute the statistics with regard to the users plan at the time.

Adding session

Another important functionality is adding of sessions. The process was similar to the previous endpoints - Creation of a parent form and then forms inheriting from the parent one. What was different was the logic behind sessions - Sessions are planned so the record corresponding to the soonest planned session is already in the database, but the data columns are not filled in (the user enters their performance of the session after they complete it) and after the date of the session is up, the session either must be marked as

```

59 def create_habit(request, habit_category: int):
60     if request.method == "POST":
61         if habit_category == HabitCategoryEnum.Exercise.value:
62             form = CreateExerciseHabitForm(data=request.POST)
63         elif habit_category == HabitCategoryEnum.Reading.value:
64             form = CreateReadingHabitForm(data=request.POST)
65         elif habit_category == HabitCategoryEnum.Learning.value:
66             form = CreateLearningHabitForm(data=request.POST)
67         else:
68             return fail_view(request=request)
69         if not form.is_valid():
70             return fail_view(request=request)
71         week_days_cleaned = [int(day) for day in form.cleaned_data['days_to_repeat']]
72         frequency = HabitsOverviewHelper().get_habit_frequency(week_days_cleaned)
73         first_rep = HabitsOverviewHelper().get_first_habit_session_date(week_days_cleaned)
74         with transaction.atomic():
75             try:|...
76
77             except Exception as e:
78                 transaction.set_rollback(True)
79                 print(f"Exception {e} occurred, traceback: {e.__traceback__}")
80                 return fail_view(request=request)
81
82     else:
83         return fail_view(request=request)

```

Figure 6.7: The endpoint for creating a Habit entity

completed by the user or it will be marked as not completed and either way new session must be planned for the date of the next repetition. Another special case is if the user enters extra session which was not planned - it was decided to keep a record of it, for computing statistics, but it does not affect the sessions schedule, if a session has been recorded and there is a not completed session planned for that day it will be considered the planned session and marked as completed, if there is no session planned on that day it will be kept in the database table but with the **is_planned** column value set to false.

■ Deleting habit

Lastly a habit must be deletable as well, this endpoint is a very simple one, it accepts only **DELETE** requests with no data and only a single parameter - **habit_id** indicating the id of the habit to be deleted. However along with the habit all Habit plans and Habit sessions must be deleted as well. There is once again an atomic transaction(32) used since if there was an error while deleting records related to each other it might threaten database integrity, even though fortunately a Habit object, binding all the other records together, cannot be deleted without deleting all other related records first due to the **Foreign Key** constraint in all of the related records, it is considered a good practice.


```

100 def edit_habit_plan(request, habit_id: int):
101     if request.method == "POST":
102         habit: Habit = Habit.objects.filter(id=habit_id).first()
103         if habit is None:
104             return fail_view(request=request)
105         if habit.category_id == HabitCategoryEnum.Exercise.value:
106             plan_model = ExerciseHabitPlan
107             form = EditExerciseHabitForm(data=request.POST)
108         elif habit.category_id == HabitCategoryEnum.Learning.value:
109             plan_model = LearningHabitPlan
110             form = EditLearningHabitForm(data=request.POST)
111         elif habit.category_id == HabitCategoryEnum.Reading.value:
112             plan_model = ReadingHabitPlan
113             form = EditReadingHabitForm(data=request.POST)
114         else:
115             return fail_view(request=request)
116         if not form.is_valid():
117             return fail_view(request=request)
118
119         plan = plan_model.objects.filter(expiration_date__isnull=True,
120                                       habit_id=habit_id).order_by('-id_order').first()
121         if plan is None:
122             return fail_view(request)
123         plan.expiration_date = datetime.datetime.now().astimezone(pytz.UTC)
124         new_id_order = plan.id_order + 1
125         if habit.category_id == HabitCategoryEnum.Exercise.value:
126             new_form = EditExerciseHabitForm(data=request.POST)
127         elif habit.category_id == HabitCategoryEnum.Reading.value:
128             new_form = EditReadingHabitForm(data=request.POST)
129         elif habit.category_id == HabitCategoryEnum.Learning.value:
130             new_form = EditLearningHabitForm(data=request.POST)
131         else:
132             return fail_view(request)
133         new_form.save()
134         return redirect('/habits/')

```

Figure 6.8: API endpoint for editing the Habit plan

6.1.3 Todos

Todos is the second of the two key components of the application alongside Habits - together these two modules are crucial for a person to plan out their tasks and other activities in their life and have them in one place.

ToDo creation

Once again the first task was to be able to create the ToDo. It required the creation of a Django Form - A Django ModelForm was used in this case, as it was the perfect tool for creating a ToDo as it is a one time activity, which, once completed, ceases to exist and since the fields are in a way defined by user (ToDo is defined by its description, due date and subtasks, which are text fields from the database point of view) there was no need to customize fields for specific types. Todos differ only by their importance level. The endpoint for the creation of a ToDo accepts only one parameter - the HTTP request itself which contains the data necessary for the Django Form. After checking that the request is sent with the right method and after checking of the validity of the data submitted a new ToDo object is created and the record is saved to the database. Upon successful creation a response with the

```

30 def delete_habit(request, habit_id: int):
31     if request.method == "DELETE":
32         with transaction.atomic():
33             try:
34                 habit = Habit.objects.filter(id=habit_id).first()
35                 if habit is None:
36                     raise Exception
37                 if habit.category_id == HabitCategoryEnum.Exercise.value:
38                     plans = ExerciseHabitPlan.objects.filter(habit_id=habit.id)
39                     sessions = ExerciseHabitSession.objects.filter(habit_id=habit.id)
40                 elif habit.category_id == HabitCategoryEnum.Reading.value:
41                     plans = ReadingHabitPlan.objects.filter(habit_id=habit.id)
42                     sessions = ReadingHabitSession.objects.filter(habit_id=habit.id)
43                 elif habit.category_id == HabitCategoryEnum.Learning.value:
44                     plans = LearningHabitPlan.objects.filter(habit_id=habit.id)
45                     sessions = LearningHabitSession.objects.filter(habit_id=habit.id)
46                 else:
47                     raise Exception
48                 plans.delete()
49                 sessions.delete()
50                 habit.delete()
51                 return HttpResponse(status=200)
52             except Exception as e:
53                 transaction.set_rollback(True)
54                 return HttpResponse(status=404)
55
56 return fail_view(request, 405)

```

Figure 6.9: Deletion of Habit and all related records

code **200** is sent back to the frontend. In case of an unsuccessful operation there is a **try - catch** structure in place to assure a smooth continuation of the application and an appropriate status code is returned. Of course firstly there was a need for the user interface to be displayed to the user and so, same as in the case of Habits, there is a central endpoint used to gather all of the information for the user requesting the loading of the page and rendering the page itself together with the data as context. Since there was no need to handle multiple request methods by one endpoint a method-based View was used instead of class-based View. Once again there is a **Helper** structure in place, so that the code is modulated and more clear. In the **ToDoHelper** there are two methods needed for this View - one queries the database for the users ToDo records and the second one sets up the form for creation of a ToDo object.

6.1.4 Users Profile

The last essential feature of the application is the Users Profile - It consists of a single form where upon displaying the page the user can see the current information kept about them such as their first name, last name, email etc. and may change this information if they wish. For this purpose there was also added a **UserInformation** table to the database, which is bound via **Foreign Key** to the **UserProfile** table and contains additional information about the user ,which might be useful for whatever purpose in the application,

such as height, weight and age.

6.2 User interface & Frontend

As was already mentioned the Bootstrap 5 CSS framework was used for styling of the user interface and the javascript XHR framework was used for client side handling of events as well as Django Templates.

6.2.1 Landing page

Creating the landing page was the first thing which was done since it is the entrance to the application and the application cannot function without it. It was elected to make it simple - just two pages where the initial page the user arrives at is the login page, to make it easier for users which are already registered, and on that page is a link leading to the second, very similar, page with the registration form.

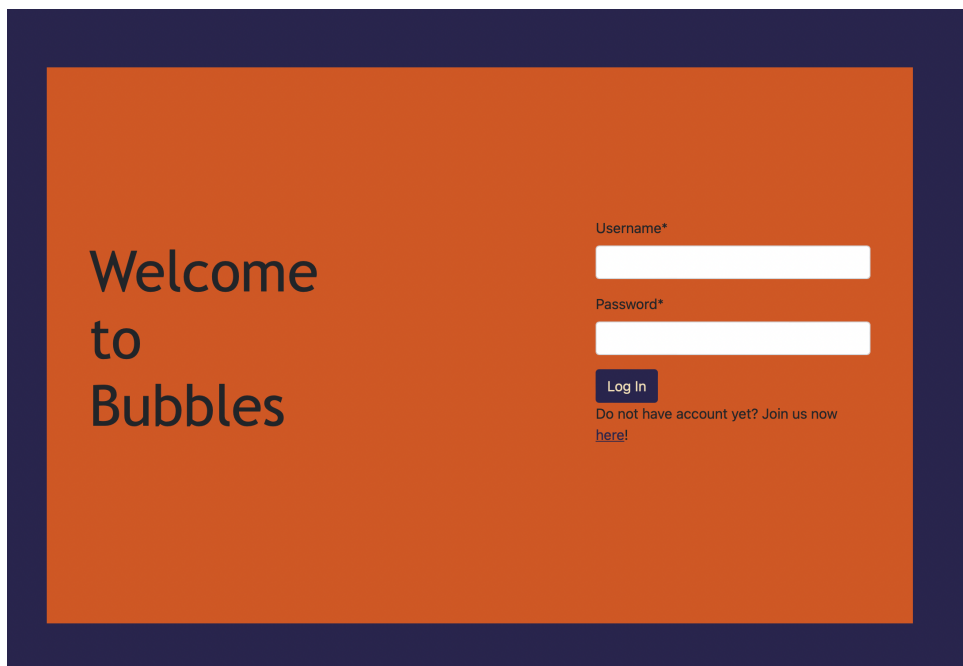


Figure 6.10: Login landing page UI

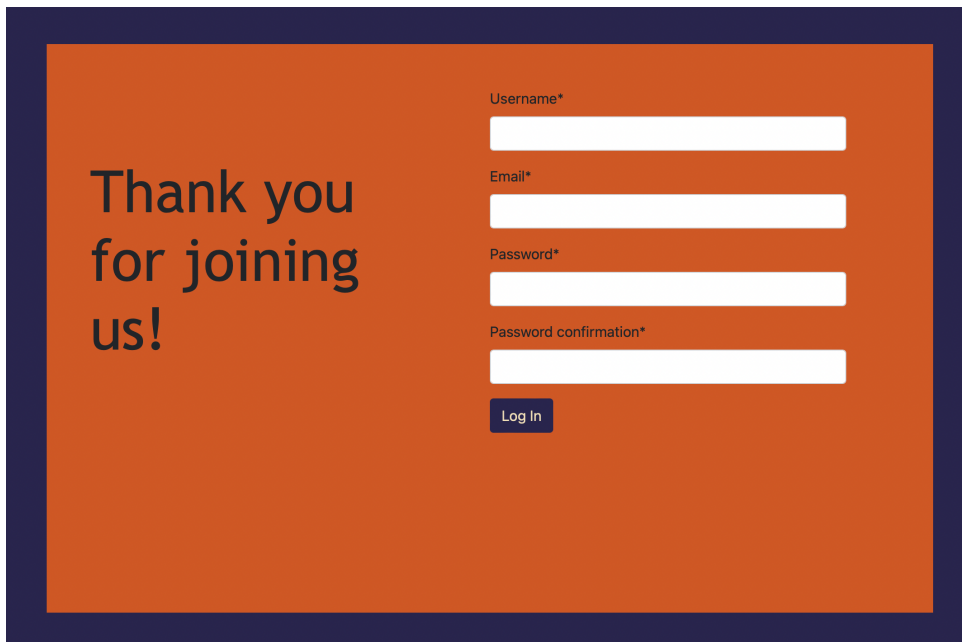
The image shows a registration form on a dark blue background. On the left, the text "Thank you for joining us!" is displayed in a large, white, sans-serif font. To the right of this text, there are four white input fields stacked vertically, each with a corresponding label above it: "Username*", "Email*", "Password*", and "Password confirmation*". Below the input fields is a dark blue button with the text "Log In" in white.

Figure 6.11: Registration page UI

■ 6.2.2 Habits page

The user interface of the Habits page had to accommodate the following functionalities for the user:

- Add a new habit
- Edit an existing habit
- Delete a habit
- Add a habit session

Furthermore it was intended to make it easy for the user to orient themselves in the page, be able to see a quick summary of their upcoming Habits (Habits with upcoming planned sessions) and overall make the page friendly to use.

■ Habit slider

An important element of the Habits page is the stripe on top of the page layout, where a cards with information about individual Habits are rendered, ordered by how soon is the next session. The Django template language⁽³⁴⁾ was used together with a very useful Bootstrap utility - Cards and Card deck⁽³⁵⁾. Upon clicking on one of these cards the Habit will render on the bottom two panels and the user can review their last session and alter their current plan.

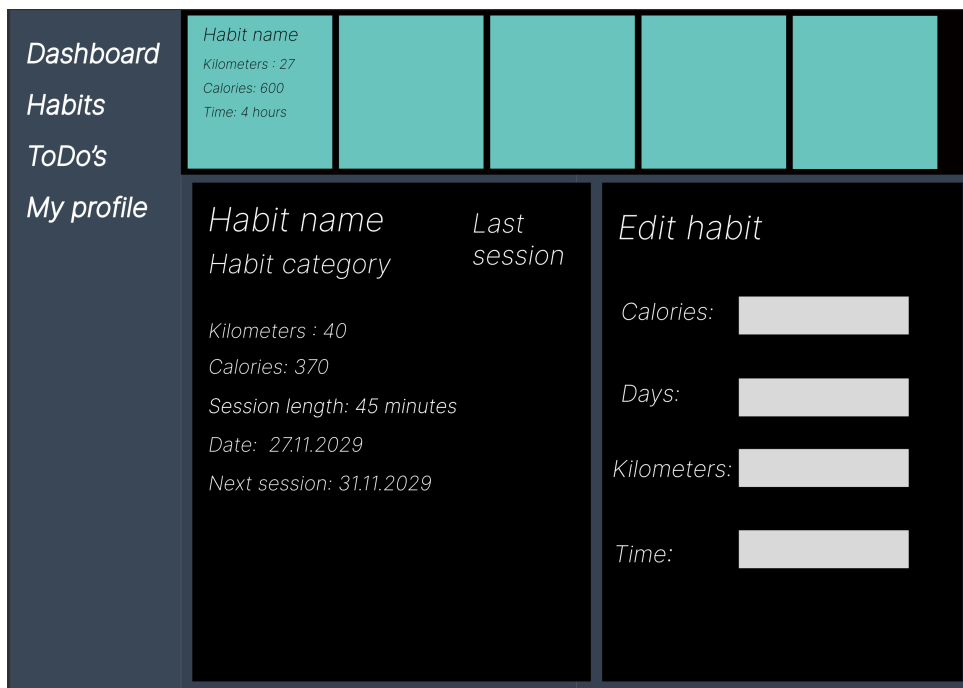


Figure 6.12: Habits page design draft

6.2.3 ToDo page

As ToDos are not editable and just a one time events, there is no need to add sessions or editing them. This is why almost all of the space on the page was used to display individual ToDos. There are three sliders, made with the use of Bootstraps Card and Card Deck elements(35) in place which differ by the level of importance of the ToDos which are contained within them. This decision was made since ToDos require little attention when it comes to inspecting them more closely and almost no administrative (such as adding sessions or editing plans) however there is a need for them to be as easily accessible as possible and all of the important information about them needs to be right on hand, when one browses them. Except for these three sliders there is one additional card which, upon clicking on it, displays a Bootstrap Modal(36) with the ToDo creation form.

Frontend Javascript XHR Framework usage

In order to keep the user experience smooth, there was a need for an inplace handling of the response from backend Views - the way Django Views work, they display the data sent from the View onto the page which sent the original request, unfortunately a way wasn't found to just return a response indicating the success of the operation, while keeping the page intact as it was with just Django alone. This is where the Javascript XHR Framework was made use of - Upon submitting the creation form the HTTP request is intercepted by the Javascript code, the data are converted to FormData type and a new

HTTP request is sent via the Javascript XHR. This is beneficial because the response from the server is returned to the Javascript code as well, where it is handled and after only a popup alert is displayed to the user - either telling them their operation was successful or that the operation was unsuccessful and asking them to try again.

■ 6.2.4 Database model

Since Django framework was used, there is already a default basic database schema which Django needs for its functionality. It consists of the following tables - **AuthGroup**, **AuthUser**, **AuthPermission**, **AuthGroupPermissions**, **AuthUserGroups**, **AuthUserUserPermissions**, **DjangoAdminLog**, **DjangoContentType**, **DjangoMigrations** and **DjangoSession**. These are tables for the Django authentication functionality (37) and other tables which Django uses. As the only table which needed direct interaction is the **AuthUser** table, the other tables were left out of the schema.. Also due to the fact, that the database schema consists of a lot of tables and that they are interconnected based on the logic behind the applications modules, the schema is presented in parts which are connected together inside these groups but the groups are not connected with one another.

■ Authentication logic

The authentication logic in this application consists of two tables - the Django table from `Django.contrib.auth` package and a custom table `UsersProfiles`. It was decided to use the Django authentication logic as there is no need for creating custom authentication application however there is the need for the ability to store some additional information about user and represent the User entity in another way and so an additional table was created which is in a 1:1 relationship with the Django `AuthUser` table and contains additional information about the user such as an email address and others.

■ Habits

Regarding habits there are three types of tables - `Habit`, `HabitSession` and `HabitPlan` where for each habit type there are tables `<HabitType>Session` and `<HabitType>Plan` since every habit type requires a different type of information which should be kept - for example an exercise habit requires keeping information about kilometers ran, calories burned etc. whereas a reading habit requires information about the title of the book which the user is currently reading. There are three types of habit tables because it is necessary to be able to differentiate between a habit session - A one time occasion where the user has performed the activity they want to make a habit of and also for the planned sessions which have yet to take place, and a habit plan, which is the plan or goal for the specific activity which the user has set for themselves. Finally the `Habits` table is for keeping all of the habits in one table and also for storing the information which has to be kept for all types of

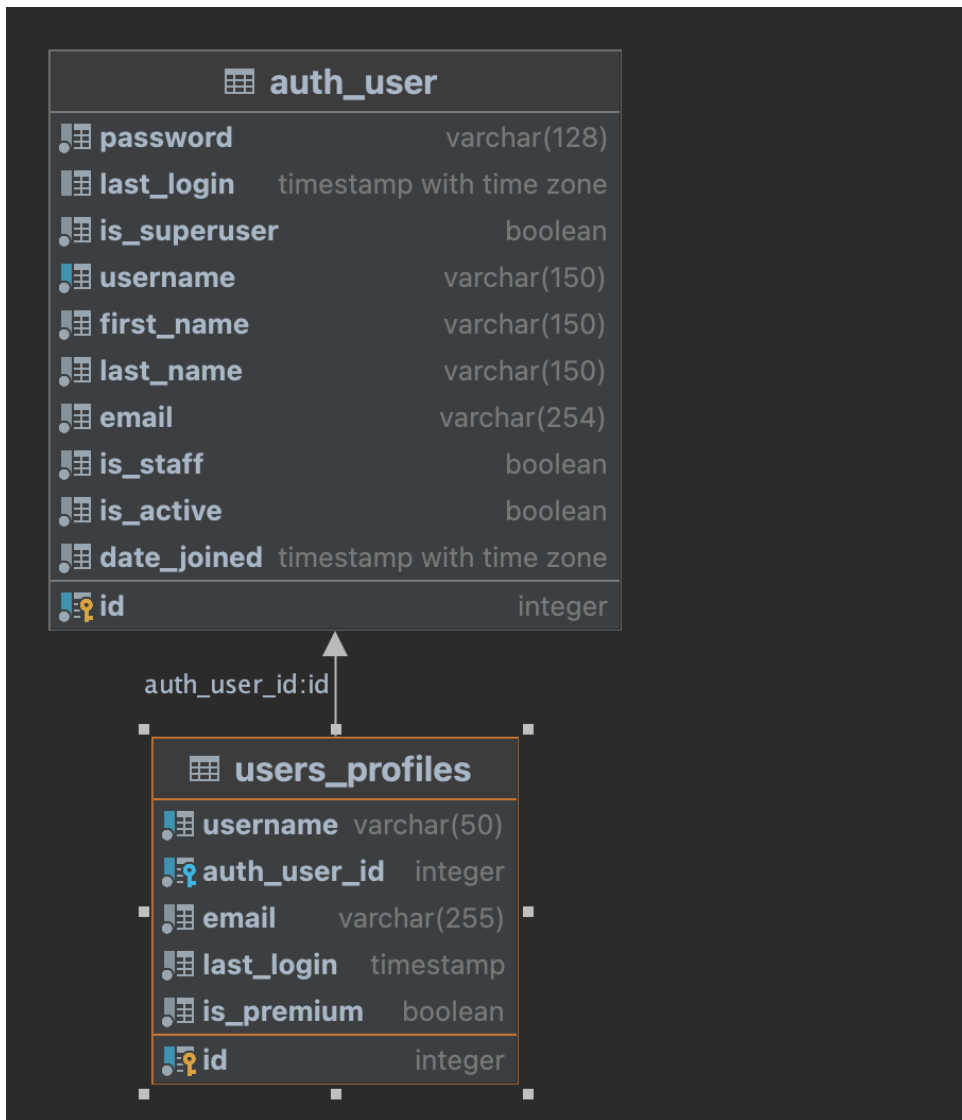


Figure 6.13: Authentication logic database schema

habits. Each habit is then bound (via foreign key to the UsersProfiles table) to one user and in the other direction each plan and each session is bound to a single habit entity. The relationships are following:

- 1:N between User entity and Habit entity respectively (each user can have multiple habits.)
- 1:N between Habit entity and HabitPlan entity (user can alter his plan for a habit, but due to the fact that there is the need to keep the information about past plans so statistics of the users progress can be made, the records are not updated but rather a new record is created for each plan change and keep information about what plan is the relevant one.)

- 1:N between Habit entity and HabitSession entity (since the habit is supposed to be an activity performed regularly over time there is the need to keep track of users actual performance in the given habit to compare it to their plan or goal.)

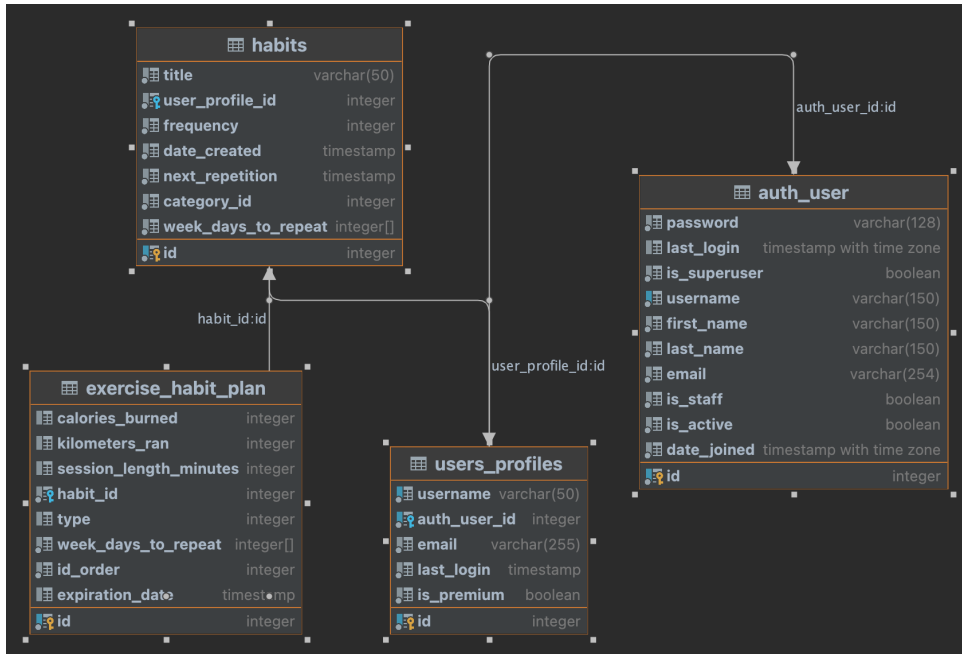


Figure 6.14: Habit database tables schema

■ Todos

Fortunately, the way the Todos module is constructed allowed for all data to be contained within one database table - Todos (the data such as description and subtasks are user-defined and so may be stored in text fields). The other information such as Todo category does not require for extra tables as well. Another fortunate thing is, that Todos are strictly one-time activities which, once accomplished, can be deleted. In reality, rather than deleting the Todo, it was elected to keep the Todos in the database but marked as done - this is so it is possible to calculate statistics regarding Todos. The Todo table contains above mentioned fields category, description and three fields for subtasks, due date - the date after which the task must be done, date of creation of the Todo, category of the Todo and finally a Foreign Key to the UserProfile table so it can be traced back to the user who created the Todo. There is only one relationship between entities and that is the **1:N relationship between the UserProfile entity and Todos**.

■ UserProfile

As it was already mentioned before, there is a UserProfile table containing additional information about the user but regarding the User Profile module in the application there is another table - UserInfo, which is not related to authentication logic, but contains other information about the user such as their weight, height, age and gender which are used to compute statistics and also change the view at the user regarding their progress (for example if a 60 year old and 25 year old users have the same performance, it might be spectacular performance for the older user but only slightly above average for the person who is at their physical peak and also it is possible to oversee weight changes of the users in time). There is once again only one relationship and that is the **1:1 relationship between the UserProfile table and the UserInfo table**.

■ 6.3 Experimental scheduler feature

Very quickly after beginning the implementation a need for some sort of mechanism which would execute some tasks periodically presented itself. In order to be able to automatically create new future habit sessions, based on the users selection of days on which they wish to repeat it, mark sessions as not completed, mark Todos as overdue and plenty other things there needs to be some sort of scheduler to perform these tasks periodically. After searching for some framework which would implement this sort of mechanism the **Celery** framework was found which is not only built precisely for this sort of functionality but has integration into the Django framework itself as well.

■ 6.3.1 About Celery

Celery is a distributed task queue framework allowing the execution of asynchronous tasks. It allows the creation of tasks and scheduling them for regular execution(38).

■ 6.3.2 Installation

For the functionality that was needed, there were three installations required - the Celery framework itself, Django-Celery-beat which is a scheduling package for the Celery framework and Rabbitmq - a message broker which Celery uses to assign tasks to its individual workers.

■ 6.3.3 Implementation

In order to run Celery framework it is necessary to set its configuration into Djangos **settings.py** file and create new python files in the core folder of the project - **celery.py** and **tasks.py** - in **tasks.py** are the definitions of tasks,

which should be executed in accordance with the schedule. Task is a python function with the decorator `@app.task`.

■ 6.3.4 Running

After starting the application itself it is necessary to firstly run the following command: `celery -A bpProject beat -l info --scheduler django_celery_beat.schedulers:DatabaseScheduler` which starts the beat scheduler for the application **bpProject** and after that `celery -A bpProject worker -l INFO` for starting a Celery worker to perform the scheduled tasks.

■ 6.3.5 Tasks implementation

There are three main problems which needed to be addressed with this scheduler - Checking for not completed Habit sessions (When the user missed a planned habit session) and sending reminders to the users every day there are some upcoming Todos or Habit sessions for the user on that day. In an ideal scenario, the checking for overdue Habit sessions would be performed every minute, however due to the overwhelming database server load and the CPU load this would generate when the number of users rises, it has been scaled down and is performed every hour, which creates sort of balance between the computational load and the need for continuous checking. As for the reminders for upcoming events, since these reminders need to be sent only once a day it has been set to be sent once every day. When checking for the overdue Habit sessions, there firstly had to be a query executed, which yields all of the Habit sessions which are overdue at the moment of checking. After that these sessions are marked as not completed and saved back and a new session is created for the date of the next planned repetition of this habit. Regarding the reminders for upcoming events - this is done for each user separately, firstly their upcoming Habit sessions and Todos are queried and then an email is drafted with a list of the upcoming events and finally sent via the Django `send_mail()` function to the email the users provided in registration.

```

92     if not todos and not users_sessions:
93         continue
94
95     email_subject = 'Bubbles: You have some activities due today!'
96     email_body = ''
97     if sessions:
98         email_body += "<h2>You have these habit sessions due today:</h2>\n<ul>"
99         for session in users_sessions:
100             email_body += f"<li>{session.habit.title} on {session.date}</li>"
101             email_body += "</ul>\n"
102     if todos:
103         email_body += '<h2>You have the following Todos due today:</h2>\n<ul>'
104         for todo in users_upcoming_todos:
105             email_body += f"<li>{todo.title} due {todo.due_date}</li>"
106             email_body += "</ul>\n"
107     email_body += "<p>Dont forget to record your activities in your Bubbles account and good Luck!</p>"
108     users_email = user.email
109     try:
110         send_mail(
111             email_subject,
112             email_body,
113             EMAIL_HOST_USER,
114             [users_email],
115             fail_silently=False,
116         )
117     except Exception as e:
118         logger.error(f"The sending of an email to user {user.username} with id {user.id} failed
119                     with the following exception: {e}")

```

Figure 6.15: The code responsible for sending emails with reminders to the users

6.4 Next steps: Modules for future development

After the key modules for the application functionality - Login & registration, Todos, Habits and Users profile, there is a lot of space for improvement and the remaining modules - Statistics section and Dashboard, will be implemented and tested in the next phase. However as these modules were all but implemented and tested, they will be presented.

6.4.1 Dashboard

The dashboard module is a useful feature of every application, as it is an overview of all users activities in one place, so the user does not need to go searching for their upcoming Todos and Habits in the individual modules, but can just quickly look in the Dashboard module and see their upcoming activities.

User interface

The user interface will use similar elements as the Habits and Todos pages - A horizontal slider of small cards, representing the individual Habits sorted by the date of upcoming sessions, so for example if the user is supposed to run tomorrow and read in two days, the former will be first in the list. Upon clicking on one of the cards a small summary of the habit will be displayed on the bottom of the page, where the plan for that habit will be available, together with a small graph representing the users activity in that particular habit for some period of time (for example for the last two weeks). The page

will also contain three larger cards, each representing the soonest Todo of each importance category and a few smaller cards, containing just the titles of other Todos which are due today.

■ Frontend

A frontend event handling will be necessary for two features - to display the specifics of a Habit after clicking on the Habit card in the Habit slider, specifically a request handler to a **GET** endpoint to collect the specific data for the Habit and also a function to dynamically change the page and display the data from the endpoint response. Another feature is the displaying of a graph under the Habit card, where the data will be received from an endpoint of the Statistics module and then will be drawn into the page via some Javascript library for drawing graphs.

■ Backend & API

The backend part of the module will surely contain a central endpoint which will return the data necessary to display all the features mentioned in the User interface section. An additional helper for this module will be helpful, as there will be a need for lot of querying with various parameters and it is a good practice to modularise this logic into smaller functions and not have the code directly in the View, where only the function calls and absolutely necessary code should be. It will also be useful to reuse functions from the Habit helper as for example a function for collecting the Habit records for a Habit slider is already implemented. Another endpoint which will be necessary is a **GET** endpoint for an individual Habit, which will return the Habit plan and other additional data, structured to be displayed in the larger Habit card.

■ 6.4.2 Statistics section

The Statistics module, while not essential, is an important part of the application functionality as it adds something more to the application than just the functionality to track ones progress - an insight into what progress are the users really making, where they might be slipping and how far have they gotten already.

■ User interface

The user interface of the Statistics module itself will be very simple - there will be one graph, or rather a window, on the page and a menu, where the user selects what statistics do they want to see (for example whether they want to see their Todo completion rate, Habit completion rate, their progress in a specific Habit or something entirely different) and specify parameters, such as from what period of time do they want the statistic to be calculated etc. However part of the UI is not just the module itself, but also a variety of

small windows/graphs which can be included elsewhere for example on the Dashboard or under a Habit card in the Habits module.

■ Frontend

The frontend should function in the following way - there will be of course the Statistics page containing the graph and menu mentioned earlier, but also there will be three templates, which will differ in size and the ability to alter the graph (for example the smallest one should be included in a Habit card and so there is no space for a menu, however the largest one which might be included in the Dashboard page might contain a small menu) which will then be included with different parameters on different places, where they will be needed.

■ 6.4.3 Backend & API

The backend API will consist of the main endpoint, which will once again provide all the data necessary for the first loading of the statistics module, however there will be also many other, smaller, endpoints which will provide different data needed for the various statistics calculation. They will be parameterized, so there isn't the need to have an endpoint for each statistic, however there will also be a limit to this, so the code stays clear. There will also be another Helper structure for the Statistics module, which will contain many general functions to query data from the database for a specific statistic for one user and then functions which will compute the statistics from this data and structure them in a convenient way for the frontend graph framework. There needs to be a very strong accent on re-usability of the code and general functions, which can be plugged together and then used in a variety of different places.

Chapter 7

Testing

In this chapter is described the process of testing of the application. There were two types of testing that took place - first type are automatic tests, which are ran each time the application is restarted or after new features are deployed, which make sure the functionality of the application is not affected in a negative way. The second type are testing scenarios for usability testing which were defined in advance and after that multiple users, who were not using the application before, were asked to perform these tasks in the application and their feedback has been collected and based on that feedback the application was either altered if some problems were recognised such as some bugs were found or some functionality was not clear, or not if no problem has been encountered in that testing scenario.

7.1 Automated testing

A series of automated tests has been created, which should be ran every time before the application is restarted, to assure that new changes did not influence the basic functionality of the application in a negative way. All of the tests were constructed in a similar manner - each test is designed to check one API endpoint and has two or more methods.

- **Setup** - The setup method creates data for all subsequent tests and if possible also creates expected outputs for the given type of input, to compare with the actual response from the endpoint.
- **Test valid** - This test sends one or more valid requests (right request type, authenticated user and valid data) to the endpoint and then collects the response from the server and checks it for validity either by comparing the response with the data created in the setup method or otherwise e.g. checks the testing database for records which are supposed to exist after this request.
- **Test invalid** - In this test case the method sends one or more invalid requests to the endpoint and then checks that the response is controlled, meaning that if an invalid output is sent, the endpoint returns appropriate

error code along with predefined error data and that the functionality of the endpoint remains unaffected.

- **Test invalid request type** - This method sends the right data to the endpoint but via either not allowed HTTP method or just different than it is supposed to be and again checks that the error response is controlled and that the functionality of the application remains unaffected.

■ 7.1.1 Example - Testing Delete Habit Endpoint

The API endpoint this test is supposed to check is the **Delete Habit** endpoint - it accepts only requests sent via the HTTP **DELETE** endpoint and only one parameter - `<habit_id:int>` ID of the habit object the user wants to delete.

■ Setup

In the setup method there is firstly the common part to all tests - the setup of testing user, who will be making this request. This user must be different for all tests so a situation is prevented, when a user would stay logged in from previous tests and by that the current user would evade the need to be logged in (It is very improbable and it shouldn't happen as the testing functionality is provided by Django itself and after each test the testing database is destroyed and a new one is created, however just to be sure, there is a new user for each testing scenario). After creating the user the **Habit** object, which is supposed to be deleted, must be created along with **HabitPlan** and one or more **HabitSession** objects, so the full range of the endpoint functionality can be tested. Lastly an ID of a not existing habit must be created, in order to test the functionality of the endpoint in case that invalid ID is sent in the parameter - As the IDs of the database rows are created via **SERIAL** postgresSQL functionality (meaning that a counter is kept and is incremented by one for every new record that is created and then its value is set as id for that record. Also to maintain database integrity, for example if an ID of a Habit object would be hard-coded at some place and so even after the object was deleted the ID would be accessible, each counter value is only used once, so for example if a 1000 Habits are created, new Habit will have ID 1001 but if 1000 Habits are created and 999 are subsequently deleted, a new Habit will also have ID 1001 so if for example the ID 2 was hard-coded in the source code no Habit object will ever be accessible by that ID ever again as it might create a confusion since it used to identify a different object, which now does not exist) the invalid Habit ID is created by increasing the highest current ID of a Habit object in the database by a very high constant, in this case 50000. This is because the invalid ID still must be valid in the sense that it must be a positive integer, as for example a -1 would be invalid ID however it would raise different problems, than this test is supposed to handle.

```

23 def setUp(self):
24     self.test_user = User.objects.create(username='test_user', password='testing_password')
25     self.test_user.save()
26     self.test_user_profile = UserProfile.objects.create(username='test_user',
27                                                       auth_user=self.test_user,
28                                                       email='test_delete_email@bubbles.xyz',
29                                                       last_login=datetime.datetime.now().astimezone(pytz.UTC),
30                                                       is_premium=False)
31     self.test_user_profile.save()
32     self.habit = Habit.objects.create(title='test_habit_delete',
33                                     user_profile=self.test_user_profile,
34                                     frequency=1,
35                                     date_created=datetime.datetime.now().astimezone(pytz.UTC).astimezone(pytz.UTC),
36                                     next_repetition=datetime.datetime.now().astimezone(pytz.UTC) + datetime.timedelta(days=1),
37                                     category_id=1,
38                                     week_days_to_repeat=[datetime.datetime.now().weekday() + 1])
39     self.habit.save()
40     self.habit_plan = ExerciseHabitPlan.objects.create(calories_burned=100,
41                                                       session_length_minutes=200,
42                                                       kilometers_ran=300,
43                                                       habit_id=self.habit.id,
44                                                       type=1,
45                                                       week_days_to_repeat=self.habit.week_days_to_repeat,
46                                                       id_order=1,
47                                                       expiration_date=None)
48     self.habit_plan.save()
49     self.session = ExerciseHabitSession.objects.create(calories_burned=100,
50                                                       session_length_minutes=200,
51                                                       kilometers_ran=300,
52                                                       habit_id=self.habit.id,
53                                                       type=1,
54                                                       id_order=1,
55                                                       date=self.habit.next_repetition,
56                                                       is_planned=True,
57                                                       completed=None)

```

Figure 7.1: Setup method of the test to prepare necessary data for the testing

Test valid

This method is supposed to test the deletion of a valid Habit object and all related objects. The test data which are supposed to be deleted are already created by the Setup method which is ran first before every testing. Firstly the testing user must be logged in to the application. This is done artificially via the **self.client.login()** method of the **TestCase** (The Django test classes should inherit from one of the existing Django classes, in this case all of the tests inherit from a **TestCase** class). After the user is logged in a request to the endpoint must be sent, this is done via the **self.client.delete()** method of the parent class (There are methods for each request method, for example for the HTTP POST method there is a method called **self.client.post()** and so on). It accepts the following parameters - the URL of the endpoint and context data - in this case no context data are required to be sent, only the **habit_id** URL parameter. After the request is sent, the response is checked for the correct HTTP status code, indicating either success or error of the operation. The check is done via the **self.assertEqual()** method of the parent class, which does assertion of equality between its two inputs, in this case that is the response code and a value 200 to check that all was successful. Furthermore, if the above assertion is passed, there is a check that no habit with the id of the habit what was supposed to be deleted exists. There is no need to check for the HabitPlan and HabitSession objects due to the **PROTECT** SQL constraint on the foreign key columns in both tables, which assure that no object may exist with foreign key value pointing to a not existing object (All deletion attempts on the referenced row will fail, unless it is no longer referenced by any row).

■ Test invalid

As was already mentioned this method is supposed to send an invalid request to the endpoint and make sure that the error response is controlled. Firstly there, once again, must be the authentication of the testing user, to make sure that the error is with the not existing object to be deleted and not with the user not being authenticated. After that a request is sent to the endpoint the same way as above, only the id sent is the invalid Habit ID prepared in the Setup method, instead of a valid ID. Finally the response code is checked again via the `self.assertEqual()` method of the parent class that it is equal to the appropriate error code, in this case **404 Not Found** indicating the resource (the Habit to be deleted) was not found.

■ Test invalid request type

Lastly, this method tests the situation where a request is sent to the endpoint via an unallowed method (in this case the endpoint only accepts DELETE requests and the test request is sent via POST method). Once again a parameter must be sent in the request since if no parameter was sent, there would be other issues this test is not designed to handle, however the check (in the endpoint) for validity of the id is after the check for method type. Upon receiving the response, there is assertion for equality between the response code and the expected **405 Method not allowed** response code.

■ 7.2 Usability testing

Even though automated testing is important and may spare a lot of time of debugging and help discover bugs before they present themselves there is also a need for testing of the entire application by unbiased users. Another positive aspect of the user testing is that, contrary to the automated tests, in user testing the application user interface and Frontend functionalities are tested as well and also the application as a whole is tested in a way that it will be used on a daily bases, unlike the automated tests which are created by the developer who created the application which is being tested as well and are sort of artificial. All of the tests bellow were done by a different user, each of which was introduced in what the application is for and what it provides, but were not given any manual to the individual application features in order to be able to discover functionalities which may not be clear to a possible future new user.

■ 7.2.1 Testing scenario 1

In the first scenario the users were registered and logged in and were told to follow the following steps:

1. Create a reading Habit with any parameters.
2. Add a session record for this Habit.

3. View details of your Habit.
4. Edit your Habit plan.
5. Delete this Habit.

■ Test result - first user

The user found the user interface intuitive, with some exceptions - they were confused about the functionality regarding displaying of the last performed session (at the time of the test only planned and completed sessions were displayed on the habits page and so if the user added a session on a day when there was no planned session, it got saved for statistics computation, but it was not displayed). Also a minor bug was discovered, when the user has no sessions yet, the data for the session are not displayed, but there are units (so for example the field where the number of calories burned is supposed to be displayed is empty, but there is the unit 'kcal').

■ Test result - second user

The user had no significant problems with the creation of the habit, they commented that the tile for creation of the habit is a significant element in the page. They also had no problem finding out how to add a session. They were a little confused on how to display the specific habit they just created (by clicking on one of the small tiles) however they quickly found out how to do it. Again they were confused about why they cannot see the details of their last session as, again, they did not plan the session for the same day and it was again saved, but not displayed. Overall they found it intuitive with some small remarks mentioned in the text.

■ Test result - third user

The third user had some struggles with the application functionality, which stemmed mostly from the fact that they did not initially understand, what is the application supposed to do and did not have a strong technical background. However after some additional explanation, they have managed to perform all of the steps in this test. Again there was a little confusion about the session not appearing on the page and so this issue will be addressed.

■ Test consequences

As a consequence of these reviews and some following internal analysis the functionality of displaying last session was changed and now the last session is displayed, regardless whether it was planned or not. Furthermore the bug discovered by the first user was fixed. Subsequently a discussion was had, about the way the application approaches the handling of Habit sessions, shortly - there may not be a need to distinguish between planned and unplanned sessions at all, since the information about the regularity with

which the user wants to have the sessions may be used purely for reminding them on these days, but there is no need to actually distinguish between planned and unplanned sessions. This idea is further discussed in the next chapter.

■ 7.2.2 Testing scenario 2

The second scenario focused on the switching between tabs and the user profile module. The user was given this list of tasks they were supposed to perform.

1. Switch to the user profile tab.
2. Add your information which you did not fill out in registration.
3. Save it.
4. Change any information in your user profile.
5. Save it again.

■ Test result - first user

As this test was a little bit simpler, the testing process went very smoothly, the user was able to perform every action on the list easily. There was one comment regarding the user interface of the User profile module, which was understandable as the UI development was not fully finished at the time, however it was noted. Another thing found was a minor bug where the internal logical values were displayed in one of the single choice categorical fields in the user profile form.

■ Test result - second user

The second user had no major issues with accomplishing all of the instructions mentioned above, they were a little confused due the bug discovered by the first user, however after a quick instruction as to what that means, there was no significant issue.

■ Test result - third user

The third user, again, had no significant issue, but too pointed out the visual appearance of the user interface in the User Profile module. They also pointed out that there is some space for extension of the User Profile module which were later discussed.

■ Test consequence

As a consequence the above mentioned bug was fixed and the user interface of the module was finished. Also the possible future extensions of the User Profile were discussed and some preliminary image of the extension of the User Profile module was established, however there are still a lot of things left to be thought out properly.

Chapter 8

Empowering the application with Machine Learning

Machine learning algorithms are widely used in many applications, to either make the user experience better and offer them insights into their data, in the case of an application similar to this it might mean analyzing their performance and offering recommendations based on this analysis, or to for example make more money from ad revenue by targeting advertising based on the user behaviour. For this analysis the problem of creating personalised workout schedules for users based on their past exercise data has been chosen. The reasons are the following:

1. Exercise habits are amongst the most popular habits for users.
2. There is a lot of data suitable for analysis and training a machine learning model in exercise habits.

8.1 Problem definition

The task at hand is to create an optimal running plan for the users based on their previous performance data, which means:

- All past running habit sessions.
- The running habit plan and its transformation in time.

Specifically, this either contains already or will contain the following data:

Past running habit sessions:

- Distance ran at each session
- Duration of each session
- Minimal heart rate at each session
- Maximal heart rate at each session
- Average heart rate at each session
- Time of the day of each session

- Date of each session
- Calories burned at each session
- Average pace or speed of each session
- Oxygen saturation at each session
- Respiratory rate at each session

Demographic data:

- Age of the user
- Weight and height of the user (BMI)
- Fitness level of the user
- Sex assigned at birth

Other data:

- User feedback to their plan
- User perceived intensity of individual sessions
-

From this information, the following features might be engineered:

- Adherence to the plan at the time of the session (for example if the user ran more or less kilometers than they set to run, if they exercised at the weekly frequency they set out to, etc.)
- Exercise session intensity for the user based on the physiological data they provided after the session.
- Running average of session completion per week (meaning how many of the planned sessions has the user missed per week)

The task is, based on the above features and data, to provide a personalised running plan in the form of the applications Habit plan - recommended number of kilometers to run, duration of the exercise, etc.

■ 8.2 Assumptions

In order to be able to solve such task, there we some assumptions required to be made, mainly in the form of the application having features it does not yet have. The application will firstly have to have integration with smart watch, otherwise it is hard to collect information like heart rate, oxygen saturation etc. Also the application will have to collect more data, than it already does, namely - user feedback to their plan and the perceived intensity of their

sessions, fitness level and also it would be beneficial not to delete the Habit history if the user is no longer interested in a Habit, since the data will be valuable for future algorithm training. All of this will also require collecting users consent to gather their data and analyse them and of course very strict anonymization policy, to make sure that the data cannot be tied to a user and used with malicious intent or stolen. Specifically not deleting users habit history will require additional informed consent, for it to be ethical (Upon wanting to delete their habit, the user will be informed that their data would be beneficial for further application development and if they consent, the data will be anonymized, but kept for further processing, unless they wish otherwise).

■ 8.3 Possible problem formalizations

There are many ways this problem could be formalized as a machine learning problem, however two of them will be discussed along with their advantages and disadvantages.

■ 8.3.1 Supervised learning approach

The problem may be formulated as a supervised learning problem, and solved by training a neural network, via the Sliding Window approach for training a neural network model(39).

■ Sliding Window

The sliding window method is most commonly used in time series forecasting problems. It is a way how to use supervised machine learning models for prediction of a next value from data, which are generated over time. There are multiple types of time series forecasting problems(40).

- **Univariate time series forecasting** - Predicting one variable at step $t+1$ from its values at time $\leq t$.
- **Multivariate time series forecasting** - Predicting two or more variables at the same time.
- **Multi-step forecasting** - Predicting more than one future step of the time series.

The process is the following - One determines the window size (meaning how many past examples they want for the future variable to be predicted for), after that the input/output pairs are created - Assume there is a multivariate variable for which the future value is to be predicted in the form of a vector of dimension N and a window of size M is chosen, then the input output pairs are a matrix of vectors which is the size of $N \times M$ where the first column of the matrix is the value of the variable at time $t-M$ and the last column is the value of the variable at time t . The output, or a target variable for

this window, is the value of the predicted variable at time $t+1$. This way a new dataset is created which consists of such matrices where the first one consists of the variable values from time 0 to time M , for the second matrix its first column is the value of the variable at time 1 and the last column is the value at time $M+1$, so if the assumption is that there is overall K samples, or measurements of the variable value, then the last data point of the new dataset would consist of vectors at time $K-M$ up to time K . This approach useful, because the individual running Habit sessions may be treated as a time series, where the session is a sample at time t . There however needs to be an alteration to this approach - as the desired output is an, in some sense, optimal plan, the next value of this series cannot be used as the target variable for the model learning so there is a need to create a dataset, specifically to label the existing data with an "optimal plan".

■ Learning setup

As was discussed above, there is a need for labeling of, potentially a very large dataset. Some of the samples of course will have to be labeled manually, however there are ways how to make the work easier. Assume there is N users, each of which has K_i exercise sessions for all i where holds $0 \leq i \leq N$. This means that in order to approach this problem as a conventional supervised learning problem there are $N \sum_{i=1}^N K_i$ samples which require labeling. Assuming that in order to have enough data, there would have to be thousands of users if not more, each of which should have at least a few dozens of sessions recorded, that means that there would be dozens of thousands if not more than a hundred thousand samples to label correctly, which is a lot of work even for a large team of people and is outright unachievable for a few people or a single person. There is however a lot of room to formulate this problem as a semi-supervised learning task. In order to simplify the problem, it will be first attempted to approach this problem as a classification problem - classify the session sequences by one of the following labels

- Increase exercise intensity considerably
- Increase exercise intensity by a little
- Don't increase or decrease the exercise intensity
- Decrease exercise intensity by a little
- Decrease exercise intensity considerably

Based on the results of this experiment, the original problem formulation will follow and it will be attempted to treat the problem as a regression problem and try to predict exact optimal values of the plan. In simplified version of the problem, the desired outcome is one of the labels, indicating how should the user alter their current performance trend and also based on the label an alternative plan will be suggested with percentage wise altered values based on the result of the classification. This simplification has the advantage of

being more achievable since the labeling of the data will require less attention than if it was supposed to be "labeled" by an optimal plan. Which means that it will be manageable to label a larger part of the dataset which will hopefully result in more quality performance of the model. The simplified proposed setup is as follows:

1. Preprocess and clean the data - normalise, remove incomplete or corrupted samples etc. (The data preprocessing will be discussed in a later section in more detail.)
2. Transform the cleaned data into stacks of column vectors (matrices as described in the previous subchapter.)
3. Manually label a part of the data (at least 5-10% or more, based on the size of the dataset) and split it into training/testing/validation subsets.
4. Create two LSTM neural networks and apply the Student - Teacher learning scenario.
5. Train the teacher network on the labeled data (this involves the entire process of training the network, tuning hyperparameters and the model architecture, measuring its performance etc.)
6. Choose a threshold of confidence for the data labeled by the teacher network (let the teacher network classify all of the remaining data and choose a subset of those, about which the network is most confident for training of the student network with the predicted labels used as soft labels)
7. Repeat the training process on the student network with the newly labeled data and again measure its performance.
8. Assess the results of this process, if the performance on the test set is satisfactory, deploy the model, if not - repeat the process or potentially if the results of the student network are not ideal, but not that bad, use it as a teacher network for another student network.

■ LSTM Network architecture

Long Short Term Memory Networks (LSTM) are networks which were introduced in the year 1997 by Sepp Hochreiter and Jürgen Schmidhuber.⁽⁴¹⁾ They were invented as a way for recurrent networks to be able to grasp a longer term context as at the time this was a task with which all the recurrent networks struggled with in practice although theoretically they should have been capable of it. Recurrent networks in general are, aside from other tasks, great fit for handling tasks related to time series - predicting the next value, classification of a series, etc. A LSTM network is structured similarly to a Convolutional Neural Network - the first few layers of the network are specialised layers, in this case consisting of LSTM cells which keep an internal memory vector for the context keeping (in case of convolutional networks the

special layers are the convolutional layers). The cells are organised into layers where each cell has the following parts:

- The cell state - the memory component of the cell which stores the memory which is updated over time.
- Input gate - Usually a sigmoid function, with a real-valued output between 0 and 1 which determines how much of the current input data should be added to the cell state.
- Output gate - Again usually a sigmoid function, only this gate determines how much from the updated internal state should be used to compute the hidden state based on the input vector and the previous hidden state
- Forget gate - Again a sigmoid function, determining how much of the previous internal cell state should be kept, based on the input layer and the previous hidden state.
- Hidden state - The output of the cell, computed from the output gate value and the updated internal cell state.

(42) As the problem at hand can be looked at as a time series related problem - the individual sessions act, in this case, as the time series and the Habit plan can be incorporated via feature engineering as an attribute of an individual time point in the series, LSTM are a great fit for such task.

■ Performance measuring

Throughout the process of learning there can arise a situation where a different metric than those mentioned here will have to be applied, however before the training process is started, these are the metrics chosen for measuring the performance of the models:

- Accuracy - accuracy is a widely known evaluation metric, in the case of multiclass classification problem it is calculated as

$$\text{Accuracy} = \frac{\sum_{i=1}^K \text{correct}_{ii}}{N}$$

where the correct_{ii} means how many samples did the network classify correctly in the i -th class. Accuracy is a good metric when the dataset is ballanced in the sense of number of samples of each class, however in the case that it is not, there is another evaluation metric - F1 score.

- F1 score - A more robust metric, good for imbalanced datasets, however may be harder to interpret.

■ Data cleaning & transformation

Before the data will be transformed the way which is described above into matrices (or stacks of column vectors) there is the need to look at the data, visualise it and perform some cleaning and transformations. Firstly it will be important to remove any incomplete data points in order to have a only quality data for the learning process or fill the missing values with for example mean value. Then it is important to normalize or standardize the continuous values and either use one-hot-encoding or embedding for some categorical values. Further there may be some features which need to be engineered - for example session consistency, heart rate reserve, BMI and other features.

■ 8.4 Goal and advantages

The goal of this task is, as has been stated above, to create an optimal exercise plan for each user, based on their past sessions and plans. The predicted plan would consist of, for example, the distance, pace, duration and frequency of repetitions based on the history of the users performance. For example - if the user has failed to adhere to their schedule for three weeks in a row, ran two times a week instead of the six times a week they set out to, and managed to ran five kilometers fewer than they planned each time, it might be beneficial to lower the plan expectations to run three times a week instead of six and to run two kilometers more than they actually ran, instead of five, which is more achievable goal, than the one they set previously but will still push them to do more and be better than they are, which will eventually lead to an improvement and over time they might even get to their original goal. The advanatages of such predictions are clear - people oftentimes incorrectly estimate themselves and their capabilities, either they set a goal which is too much for them and after then inevitably fail to live up to their high expectations, they quit trying altogether. Or, on the other hand, people underestimate themselves and the challenge they set for themselves is too easy and they do not feel their progress, as they will not be making any, if they do not push themselves over their comfort limit. Of course such predicted plan is not flawless and needs to be presented more as a recommendation rather than a professional advice. This is of course why the users feedback must be incorporated and the plan must be changeable, if the situation asks for it. However it is a very available way how to estimate ones strengths and weaknesses and advice them on their way to a more active life. From the perspective of the application this is a great way to get users to stick with the application and use it and avoid users leaving not because there are problems with the application, but rather because they feel like they did not achieve what they wished to achieve.

Chapter 9

Conclusion

This chapter is a presentation of the intended uses of this application, discussion of the benefits of this application and also the discussion of possible extensions in functionality of the application as well as conclusion and final assessment.

9.1 Looking back: An assessment of the process

The process started with an analysis of the domain itself, from a psychological perspective, various psychological concepts were discussed and looked into more closely. As a result of this analysis the first idea of the philosophy the, at the moment, future application should uphold was created. After this analysis there was a close look and analysis of other applications from the domain of personal development and habit creation, to draw inspiration and to look at the possible mistakes of others to be able to avoid them. After acquiring a glance on the psychological point of view of this domain and seeing how others have attempted to tackle the issue of creating a personal development application which would truly help people reach their goals and better their lives there was a formalisation of the, at that time, more abstract concepts of how the application should look like and what should its main functionalities be. After specifying and formalising the functional and non-functional requirements, another analysis took place, this time into possible technologies upon which the application would be built and the best suited programming languages and frameworks were selected in order to achieve the best results. When this solid philosophical and technical foundation was laid, the implementation of the application itself took place, which is carefully described in the previous chapters. A high priority was placed on trying to explain the thought processes of the author regarding the individual implementation choices. Finally there was a discussion of possible enhancements of the application via Machine Learning and a structure of the next steps in this direction was set. The process of implementation of such functionality is one of the next highest priorities in future development as well as extending the application functionality in other ways, which are described in this chapter. Looking back there was a lot to learn and there were some steps which from the current point of view might have been tackled

in a different way. This has helped to shape the authors view at the future steps and the author will reflect this in the continuation of this process and an emphasis will be put into not doing the same mistakes twice.

■ 9.2 Usage & benefits

As must be clear at this point, the main point of the application is to make it easier for the users to organise their lives, keep track of their Habits and Todos and create new ones in an organised manner. Specifically the Users can create Habits, choose its type and based on that type track their progress and record their performance. Also they can setup a goal, or more precisely a benchmark to which they want to perform in the form of Habit plan, which they can later edit and change the values of either set a harder goal as they get better or lower it in case its previous value was too high. Furthermore the users may track their Todos which may be day-to-day tasks such as to take out the trash or pick up milk from the store and others or it may be for example an important event such as a birthday of a close person or a homework deadline. These Todos also have an importance level and are separated and grouped together based on this importance level. Also thanks to the automatic scheduler functionality users get reminders for their events via an email notification on the day these events take place and do not need to worry about having to plan their individual habit sessions as they are planned for them based on the schedule provided. On the other hand the users are also 'kept responsible' if they do not attend their sessions thanks to this scheduling functionality since their sessions are regularly checked and if they are overdue, they are marked as not completed, so they can count on having accurate statistics, which may not be pleasant at all times, but it gives an important realistic feedback this way.

■ 9.3 Possible functionality extension & upgrades

In an application such as this, there is always not only a room for improvement, but also many functionalities which might make it better and extend its functionality. Very often developers get the idea for such extensions after they have started developing the application and they can actually see what they previously only planned. This section is a presentation and a discussion of such ideas which might make the application easier to use, extend its functionalities or make it overall better.

■ 9.3.1 Messaging inside the application

The first possible improvement that comes to mind are reminders which would be in the form of a message inside the application which might remind the user that their events are about to take place, introduce them to future updates and the new features that would come with it or allow them to message other

users. The lastly mentioned possible usage of such functionality might be extended to a fully functional social network inside of the application and it is discussed in the upcoming subsection. Regarding the messaging from the application and its developer team to its users, on one hand it might not serve the purpose of luring the users to open the application as an email might, since in order to get the message they would need to open the application first by themselves, however this way the current information that is sent to the user via email might be delivered to them via the in-application message and instead of a long email, they would only receive a notification via email, that they have some new messages in their application inbox. This way the users would get reminded of the application via an email, so this purpose would be kept, but they could read the news in a controlled environment of the in-application internal messaging system and there would be smaller chance, that the information will be lost in a tidal wave of emails every person receives every day.

■ 9.3.2 An internal social network

There are many arguments for such an extension for an application such as this, for example:

- A social network might increase the attractiveness of the application towards potential new users and also to current users who are not sure whether they wish to keep using it.
- If implemented well with strong social and competitive features it might improve the user experience and use competitiveness of users to push them to better results.
- Might also improve the user experience in a way, that the application would create and post challenges which the users might pledge to and besides from other factors, it might guide the users if they had no idea how to begin with their journey.

A social network inside the application is an idea which is used in many applications. The user profile module might be extended to serve as a face of the user in the application, or rather in the social network, show their successes and recent achievements. Of course the inseparable part of any social network is a social feature, where users might communicate with each other, create groups, social events (for example in the form of challenges, where the users might sign up for a challenge and meet with each other in order to fulfill it) and connect via a friend list functionality. If the users were friends in this social network, then they might see more detailed information about each other such as what the other user is currently doing, how are they doing and their recent achievements.

■ 9.3.3 Achievements & streaks

In order to motivate people better a new functionality of Achievements and streaks could be introduced - Achievements are sort of milestones, each person can reach and get a reward for it. Streaks would express for how long the user has managed to stick to their plan or schedule. This would also be very well integratable with the social features as users would have their public profile where these achievement badges might be displayed and it would also be a great opportunity to implement gamification features into the application. For example - each achievement would equal some amount of points added to the users avatar, for which they might unlock some type of either bonus content or customize their application avatar.

■ 9.3.4 Public API

A very useful feature for users who want either to control what data is the application collecting about them, or would like to have the data they generated by for example exercising for some personal project. Many applications offer this feature, one of them is for example Habitica(13). Essentially there would have to be created a public API which would be accessible for anyone with an API key and they could pull their data via HTTP requests and for example analyse their performance or try to recognise some patterns in their behaviour or do whatever they wish to do.

■ 9.3.5 Extend habit type range

The available types of habit are very limited, which is due to the fact that the application is still more of a prototype rather than fully finished product. There is a lot of room to extend the types of habits from which the users can choose for, add types of information the users can store about their habits (type of information means for example the duration of session etc.). Especially the exercise habit type could be extended for example with many forms of exercise and sports and according fields for each type, for example for a weight lifting habit the users might record the exercises they did and number of repetitions and the weights they exercised with. Other important habit type, which is not available at the moment, but has a lot of promise and wide popularity amongst users of this type of application, is the recording of meals the user eats each day. A database of meals and individual groceries could be made, from which the users might select what they ate and what amount and subsequently the application could calculate their daily calorie intake and their macro nutrients which would greatly synergize with the tracking of their exercise activity.

■ 9.3.6 UI facelift

As the application in its current form is just a prototype, the user interface has a lot of space for improvement. Both the design and the user experience

could be improved. In hindsight since, as is stated in the subsection above, there is a lot of room for additional habit types and overall information to track, the current form of displaying habits seems not well extendable. The habit slider which is present at the moment could be kept, but each habit would have an individual card (page) where all of this information would be more clearly displayed and even additional information, computed from the basic data about the habit, could be computed.

■ 9.3.7 Integration with smart watch

Integration with smart watch is a very popular feature and today feels almost necessary for such an application to succeed. Of course there is a lot of information the user would still have to manually input, however by connecting ones smart watch with the application, exercise data such as minimal, maximal, average heart rate, calories burned, duration of exercise, type of exercise and many more could be automatically parsed into the application, the session could be created without users effort and they could just add some information which cannot be parsed from the watch later.

■ 9.3.8 Finishing the implementation of other modules

Before all of the above, there is still a step which will be taken - finishing of the implementation of the Statistics module. As was mentioned in early chapters, statistics of ones progress are a great motivators and can also provide insight into ones performance and allow an analysis of reasons why is one not performing to the standard they set for themselves or maybe can provide additional motivation after seeing they are doing great and will be pleased with themselves which will promote further development. The implementation phase is already described in the implementation chapter, however it is grasped from a more technical point of view, however the specific statistics which will be present are the following:

- Session completion rate for each habit and overall habit completion rate per period of time (week/month)
- Todo completion rate per week/month
- Various averages over periods of time (such as average kilometers ran per session/week/month)
- Current trend analysis in various habit parameters, which can be looked at as a series of numbers (Calories burned, kilometers ran, pages read, ...)

After the statistics section there will also be a simple Calendar module inside the application, which will automatically contain users activities (Todos, Habits and custom events). It will be done via integration with users preferred calendar server (there will be several choices to choose from) or, in case the user either does not wish to or does not use any such calendar service at all,

there will be a default integration with a Google Calendar where there will be a new calendar created for each user. The reason for using integration of third party services is firstly comfort - many users use various calendar services and this way they can easily incorporate their personal growth into already known environment. As for the default use of a Google Calendar service - there is no need for custom calendar in this application as the functionalities required are very simple - to create an event, add repetitions, reminders and see when is the activity taking place, and since there are many, already existing and well implemented, solutions the author sees no need for a custom calendar service.

Appendix A

Project structure

Bellow is the directory diagram of the enclosed project, which is the result of the implementation process, with description of individual files functionality.

```
/bpProject - Main folder encapsulating the entire project
├── base - Base folder, containing Enums and a BaseModel
│   ├── enums - Folder containing all Enum classes used across the project
│   │   ├── codebook_category_enum.py - Enum containing categories
│   │   │   of Enums for easier DB value retrieval
│   │   ├── exercise_type_enum.py - Enum containing the possible types
│   │   │   of exercise
│   │   ├── habit_category_enum.py - Enum containing the possible
│   │   │   habit object types
│   │   ├── habit_repetition_enum.py - Enum containing the possible
│   │   │   habit repetition frequency values
│   │   ├── learning_subjects.py - Enum containing the possible learning
│   │   │   subject values
│   │   ├── todo_importance_enum.py - Enum containing the possible
│   │   │   types of todo importance levels
│   │   └── week_days_enum.py - Enum containing the individual days
│   │       in a week
│   └── base_model.py - Base model; a predecessor for all models
│       (unused)
├── bpProject - A main folder of the Django project
│   ├── __init__.py - An init class of this module
│   ├── asgi.py - One of Django's configuration files
│   ├── celery.py - Configuration file for the Celery framework
│   ├── settings.py - Django settings file
│   ├── urls.py - Django file for defining routes
│   └── wsgi.py - Django wsgi configuration file
├── dashboard - Prepared folder for the Dashboard module; Contains
│   only default auto-generated Django files
├── habits - Folder containing the Habits module
└── helpers - Folder containing the helper classes for the Habits
    module
```

- └─ habits_overview_helper.py - A helper class for the Habits module; contains many utility functions
- └─ migrations - Django folder containing files necessary for Django migrations
- └─ __init__.py - Django generated init file for this module
- └─ admin.py - Django generated file
- └─ apps.py - Django generated file
- └─ forms.py - Python file containing the Forms of the Habits module
- └─ models.py - Python file containing the Models of the Habits module
- └─ tasks.py - Python file containing the definition of Celery tasks for this module
- └─ tests.py - Python file containing the automated tests for this module
- └─ views.py - Python file containing the Views (API endpoints) for this module
- └─ login - Folder containing the Login and registration modules
 - └─ migrations - Folder containing the Django migrations files for this module
 - └─ __init__.py - Django generated init file for this module
 - └─ admin.py - Django generated file
 - └─ apps.py - Django generated file
 - └─ forms.py - Python file containing the Forms for these modules
 - └─ models.py - Python file containing the Models for these modules
 - └─ tests.py - Python file containing the automated tests for these modules
 - └─ views.py - Python file containing the Views (API endpoints) for these modules
- └─ static - Folder containing the static content of the application
 - └─ css - Folder containing .css files generated by Bootstrap 5 framework
 - └─ icons - Folder containing icons used in the application
 - └─ js - Folder containing javascript files used in the application
 - └─ custom_js - Folder containing custom .js files for the application
 - └─ add_habit_popup.js - JS script controlling the interaction of creating a new Habit
 - └─ add_todo.js - JS script controlling the interaction of creating a new Todo
 - └─ habit_record_list.js - JS script controlling the dynamic content changes in the Habit page
 - └─ todo_sliders.js - JS script controlling the interaction of deletion of a Todo
- └─ templates - Folder containing all HTML templates of the application

- habits - HTML templates for the Habits page
 - habits.html - Root HTML file for the Habits page
- login - HTML templates for the Login and Registration pages
 - login.html - Login page HTML template
 - registration.html - Registration page HTML template
- modals - HTML templates for various components, which can be included in multiple places
 - todo_modals - HTML templates for modals of the Todo page
 - add_todo_fake_card.html - HTML modal for invocation of the modal for creation of a Todo
 - create_todo_popup.html - HTML modal for creation of a Todo
 - fake_todo_card.html - A placeholder displayed if no Todos have been created yet
 - todo_card.html - A template for a single Todo item
 - todo_slider.html - A modal used for displaying a row of multiple Todo cards
 - add_habit_fake_card.html - A modal for invocation of creation of a Habit
 - add_habit_popup.html - A modal for creation of a Habit session
 - base_page_with_navbar.html - A root template for all pages with navigation bar
 - create_habit_popup.html - A popup modal for creation of a Habit
 - default_edit_habit_plan.html - The initial modal showing the current habit displayed upon loading the page
 - delete_habit_popup.html - A popup for the deletion of a Habit (confirmation)
 - edit_habit_plan.html - Modal form for editing of a Habit plan
 - error_page.html - An error page shown in case of an error in the application
 - habit_card_basic.html - A template for the Habit card on top of the page
 - habit_main_content.html - A template for the entire Habits page; included in habits.html
 - habit_record_list.html - Modal for showing multiple Habit cards on top of the page
 - navbar.html - Template containing the navigation bar
- registration - Legacy folder, in which there initially have been the templates for registration page (unused)
- todos - Folder containing the Templates for the Todo page
 - todos.html - Root template for the Todo page

- └─ user_profile - Folder containing templates connected with the User Profile module
 - └─ user_profile.html - Template for the User Profile page
- └─ todos - Folder containing the Todos module
 - └─ helpers - Module containing the Helper classes for this module
 - └─ todos_helper.py - The helper class for the Todo module; contains many utility functions
 - └─ migrations - Django generated folder with files connected with Django migrations
 - └─ __init__.py - Django generated init file
 - └─ admin.py - Django generated file
 - └─ apps.py - Django generated file
 - └─ forms.py - File containing the Django Forms for the Todos module
 - └─ models.py - File containing the Django Models for the Todos module
 - └─ tests.py - File containing the automated tests for the Todos module
 - └─ views.py - File containing the Views (API endpoints) for the Todos module
- └─ user_profile - Folder containing the User Profile module
 - └─ migrations - Django generated folder with files connected with Django migrations
 - └─ __init__.py - Django generated init file
 - └─ admin.py - Django generated file
 - └─ apps.py - Django generated file
 - └─ forms.py - File containing the Django Forms for the User Profile module
 - └─ models.py - File containing the Django Models for the user Profile module
 - └─ tests.py - File containing the automated tests for the User Profile module
 - └─ views.py - File containing the Views (API endpoints) for the User Profile module
- └─ celery.log - Log file for the Celery framework

Appendix B

Application visual appearance

Bellow are Figures representing the visual appearance of the application. Please note that the application working title is "Bubbles", and it is the reason this name is mentioned on several places in these figures.

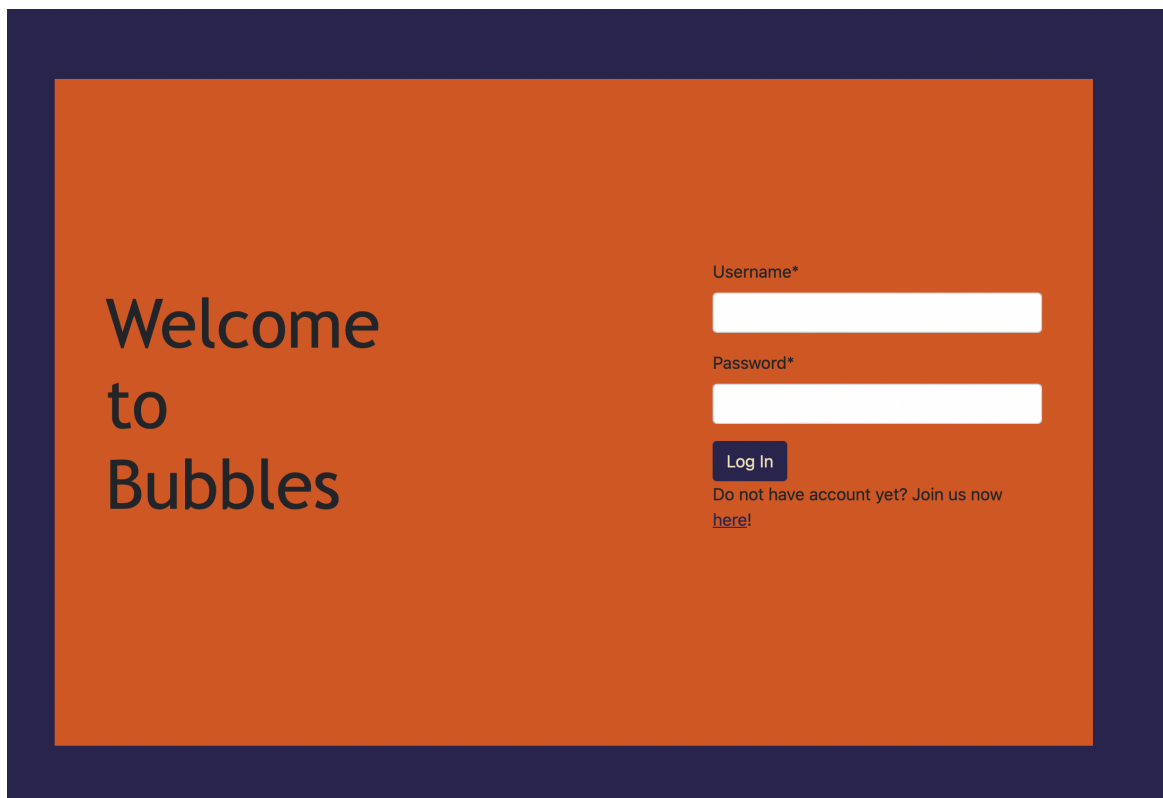


Figure B.1: Login page of the application.

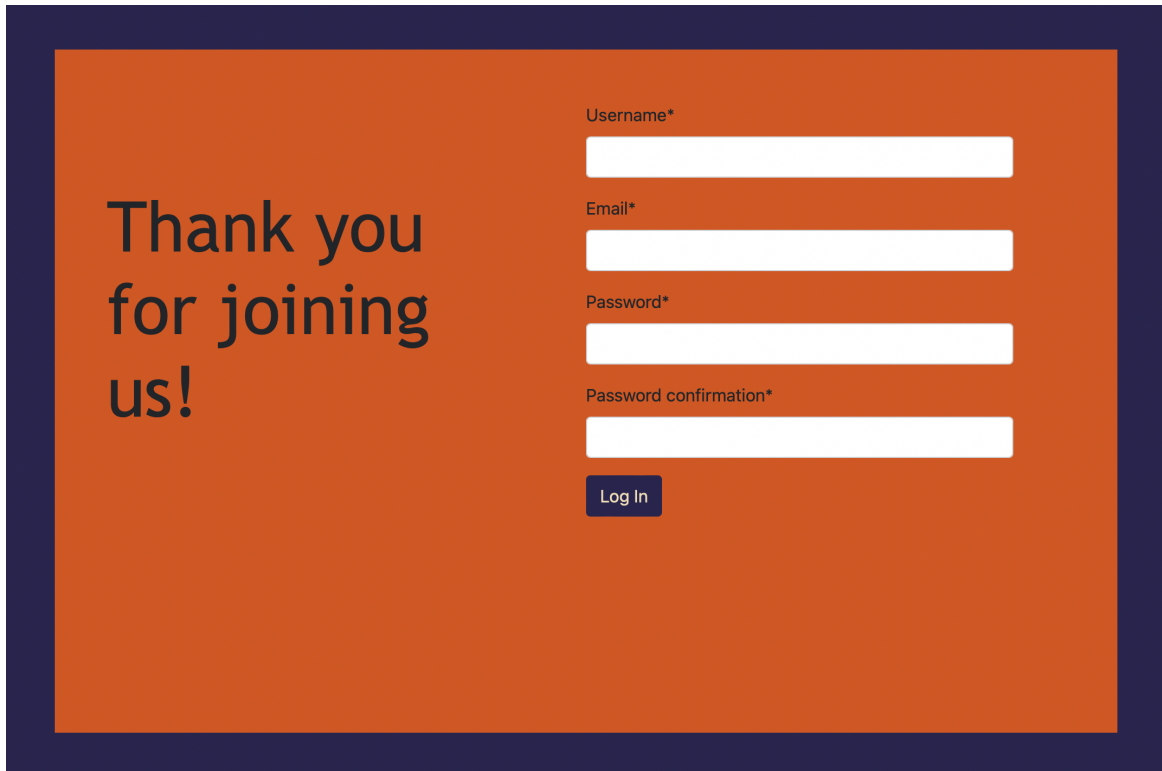


Figure B.2: Registration page of the application.

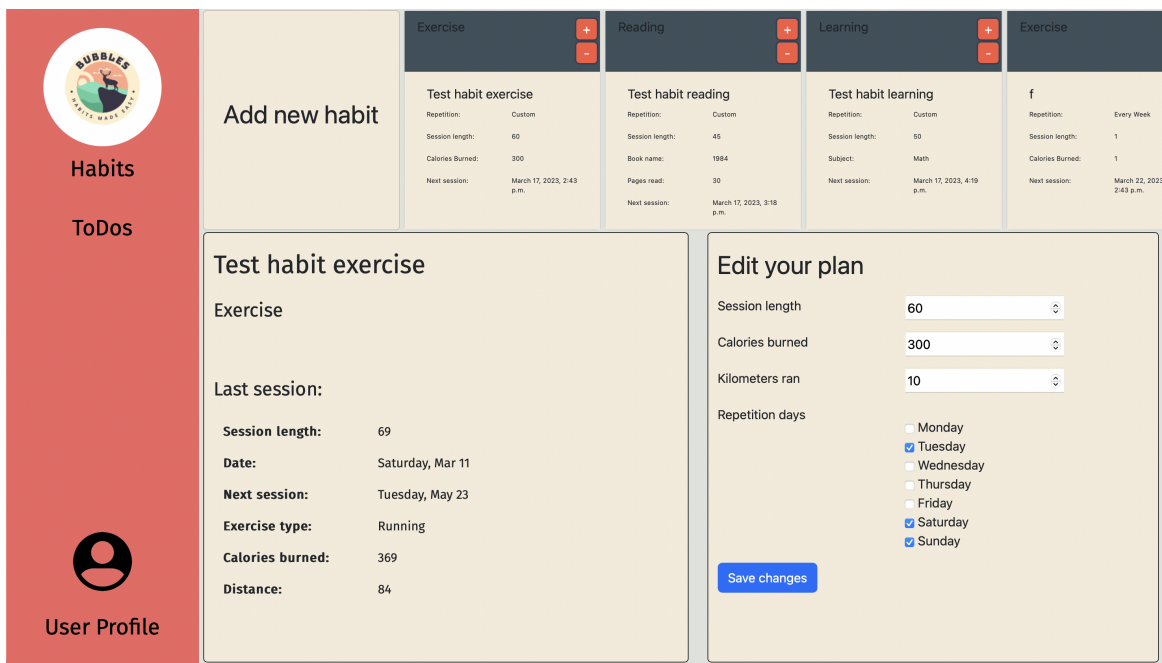


Figure B.3: Habits module page.

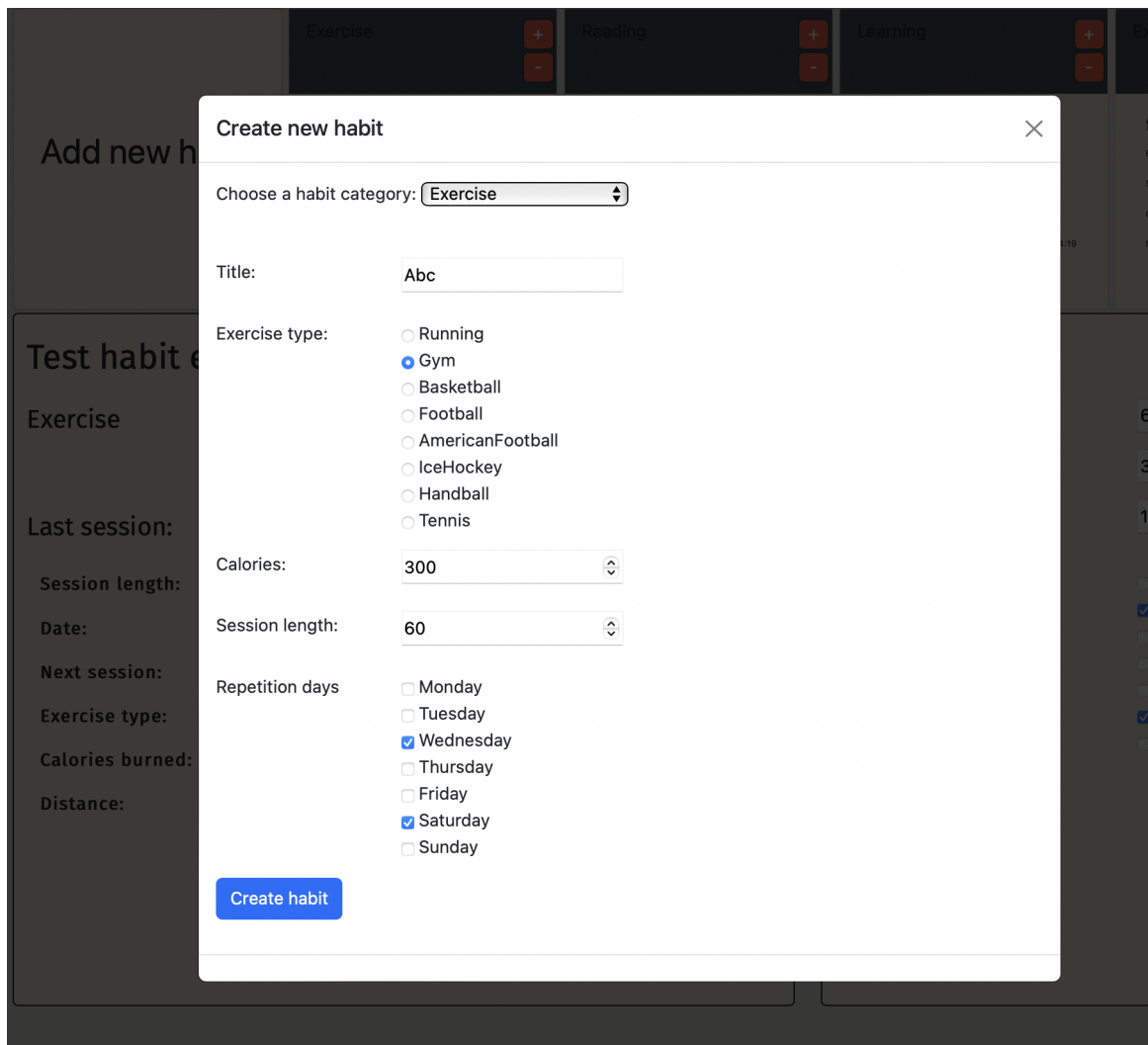


Figure B.4: Habits module page modal for creating new Habit.

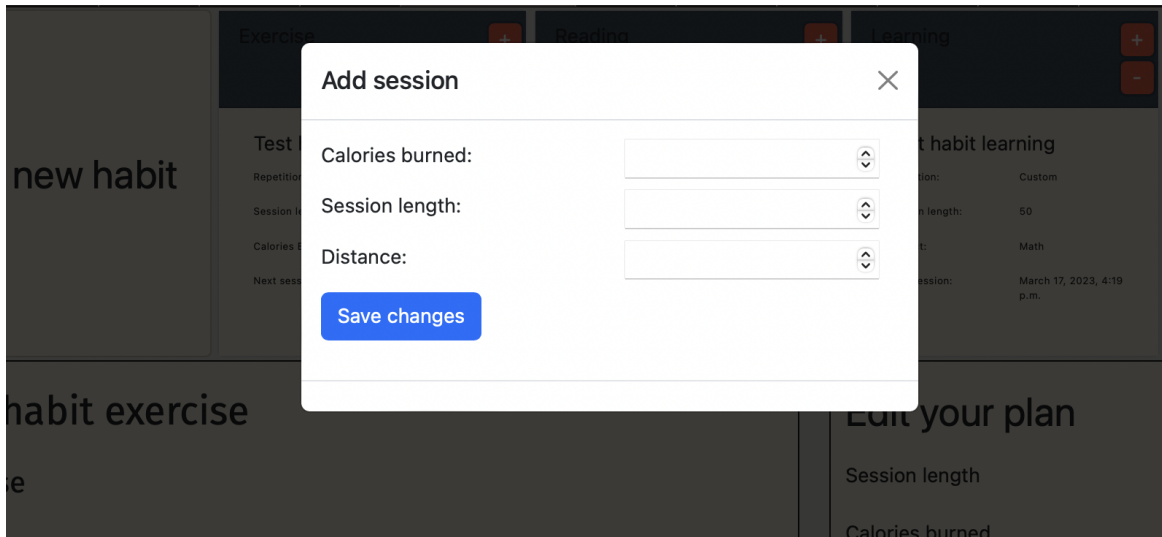


Figure B.5: Habits module page modal for adding a new Habit session.

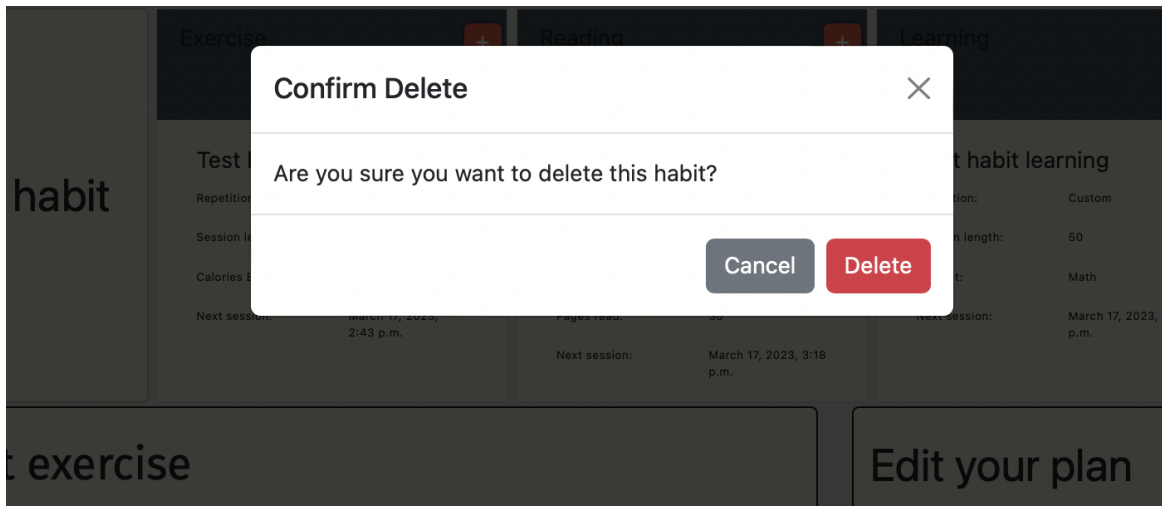


Figure B.6: Habits module page modal for deleting a Habit

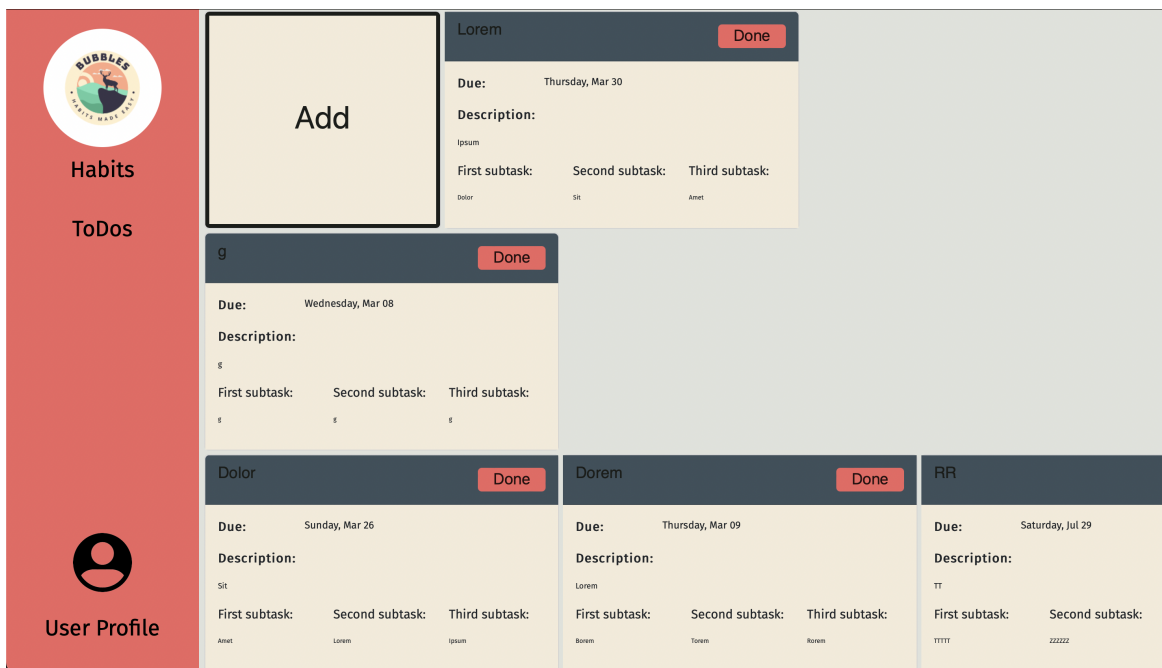


Figure B.7: Todo module page.

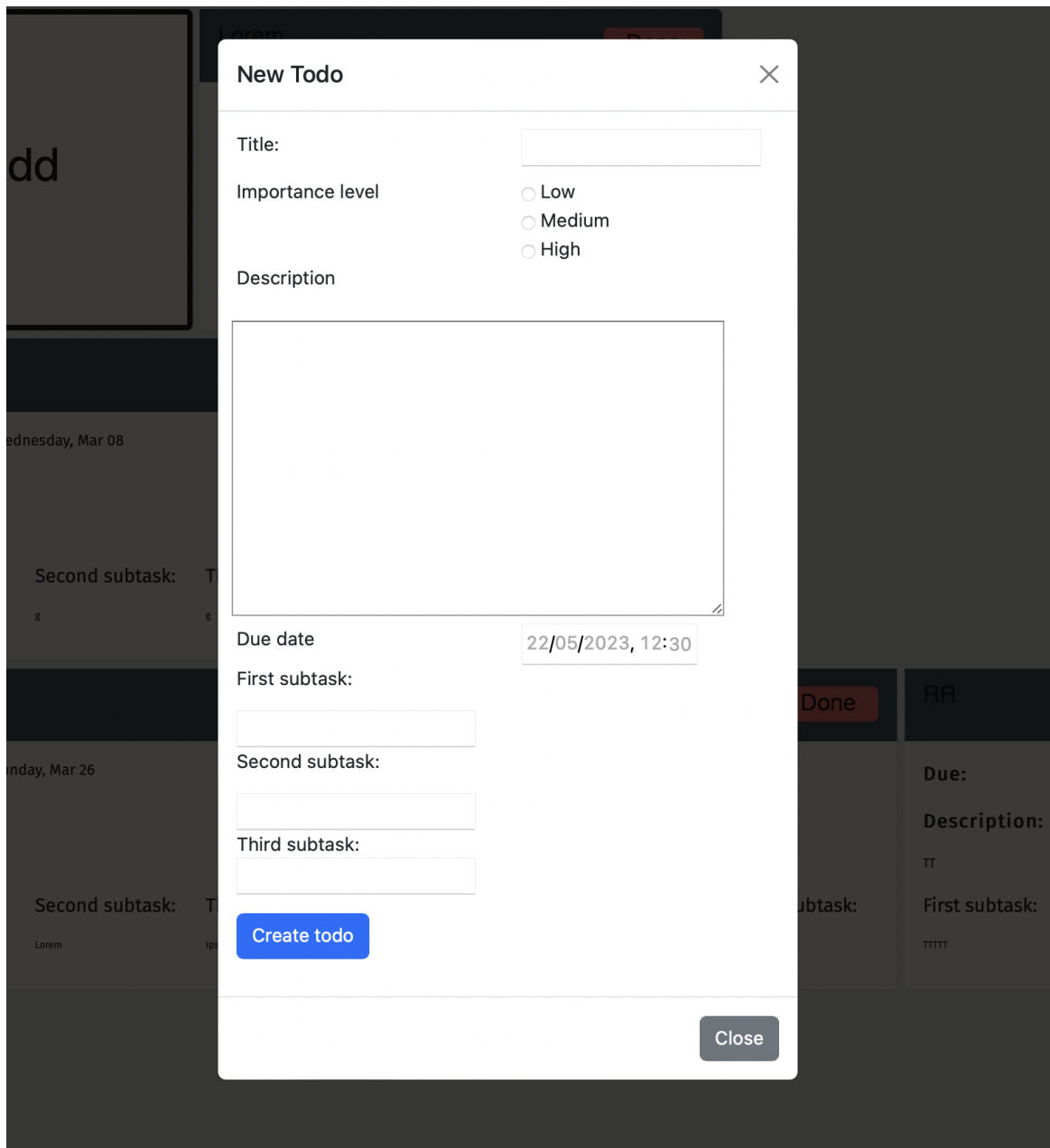


Figure B.8: Todo module page modal for creating a new Todo.



Bibliography

- [1] R. M. Ryan and E. L. Deci. (2000) Self-determination theory and the facilitation of intrinsic motivation, social development and well-being. Accessed: 2022-10-01. [Online]. Available: https://www.researchgate.net/publication/11946306_Self-Determination_Theory_and_the_Facilitation_of_Intrinsic_Motivation_Social_Development_and_Well-Being
- [2] Intrinsic motivation definition. Accessed: 2022-10-28. [Online]. Available: <https://www.frontiersin.org/articles/10.3389/neuro.12.006.2007/full>
- [3] Definition of extrinsic motivation. Accessed: 2022-10-02. [Online]. Available: <https://www.sciencedirect.com/topics/psychology/extrinsic-motivation>
- [4] Definition of habit and explanation of why habits exist. Accessed: 2022-10-07. [Online]. Available: <https://psyarxiv.com/kbpmv/>
- [5] How bad habits connected with pleasure and a dopamine release are harder to get rid of. Accessed: 2022-10-12. [Online]. Available: <https://newsinhealth.nih.gov/2012/01/breaking-bad-habits>
- [6] A text about how the 21 days rule to create a habit has been created and why it is not true. Accessed: 2022-10-15. [Online]. Available: <https://jamesclear.com/new-habit>
- [7] J. Clear, *Atomic Habits*, 2018, accessed: 2022-09-29.
- [8] Maslow's hierarchy of needs. Accessed: 2022-10-11. [Online]. Available: <https://www.simplypsychology.org/maslow.html>
- [9] Personal development definition. Accessed: 2022-10-26. [Online]. Available: <https://ukcpd.co.uk/personal-development/what-is-personal-development/>
- [10] List of similar applications, their advantages disadvantages. Accessed: 2022-09-28. [Online]. Available: <https://zapier.com/blog/best-habit-tracker-app/>

- [11] List of similar applications, their advantages & disadvantages. Accessed: 2022-09-28. [Online]. Available: <https://en.softonic.com/top/personal-development-apps>
- [12] List of similar applications, their advantages & disadvantages. Accessed: 2022-09-28. [Online]. Available: <https://www.makeuseof.com/self-improvement-android-apps-unlock-potential/>
- [13] About habitica application. Accessed: 2022-10-31. [Online]. Available: https://habitica.fandom.com/wiki/What_is_Habitica%3F
- [14] About way of life application. Accessed: 2022-11-05. [Online]. Available: <https://apkcombo.com/way-of-life-habit-tracker/com.wayofflife.app/>
- [15] About habitify application. Accessed: 2022-11-06. [Online]. Available: <https://www.dailyhabits.xyz/habit-tracker-app/habitify>
- [16] Self determination theory definition. Accessed: 2022-10-01. [Online]. Available: <https://selfdeterminationtheory.org/theory/>
- [17] About mvc software architectural pattern. Accessed: 2022-12-02. [Online]. Available: <https://www.codecademy.com/article/mvc>
- [18] About mvp software architectural pattern. Accessed: 2022-12-02. [Online]. Available: <https://stackoverflow.com/questions/141912/alternatives-to-the-mvc>
- [19] About django framework. Accessed: 2022-12-02. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>
- [20] About springboot web application framework. Accessed: 2022-12-04. [Online]. Available: <https://www.digitalocean.com/community/tutorials/key-components-and-internals-of-spring-boot-framework>
- [21] Comparison of postgresql and mysql databases. Accessed: 2022-12-04. [Online]. Available: <https://kinsta.com/blog/postgresql-vs-mysql/>
- [22] About vuejs frontend framework. Accessed: 2022-12-04. [Online]. Available: https://linuxhint.com/about_vue_js/
- [23] Pros and cons of vuejs frontend framework. Accessed: 2022-12-04. [Online]. Available: <https://ddi-dev.com/blog/programming/the-good-and-the-bad-of-vue-js-framework-programming/>
- [24] About reactjs and its comparison to vuejs. Accessed: 2022-12-04. [Online]. Available: <https://hygraph.com/blog/vuejs-vs-react>
- [25] About bootstrap, bulma and other css frameworks. Accessed: 2022-12-17. [Online]. Available: <https://www.browserstack.com/guide/top-css-frameworks>

- [26] About bootstrap, bulma and other css frameworks. Accessed: 2022-12-17. [Online]. Available: <https://geekflare.com/best-css-frameworks/>
- [27] About bootstrap and other css frameworks. Accessed: 2022-12-17. [Online]. Available: <https://www.uplers.com/blog/what-are-the-pros-cons-of-foundation-and-bootstrap/>
- [28] About django security measures and why is it secure. Accessed: 2023-03-30. [Online]. Available: <https://docs.djangoproject.com/en/3.2/topics/security/>
- [29] Installing pyenv and creation of python virtual environment. Accessed: 2023-02-15. [Online]. Available: <https://jordanthomasg.medium.com/python-development-on-macos-with-pyenv-virtualenv-ec583b92934c>
- [30] What is python virtual environment. Accessed: 2023-02-15. [Online]. Available: <https://www.geeksforgeeks.org/python-virtual-environment/>
- [31] Guide on how to create a login and registration page in django. Accessed: 2023-02-16. [Online]. Available: <https://ordinarycoders.com/blog/article/django-user-register-login-logout>
- [32] Django database transactions. Accessed: 2023-02-20. [Online]. Available: <https://docs.djangoproject.com/en/4.1/topics/db/transactions/>
- [33] About django modelform class. Accessed: 2023-02-19. [Online]. Available: <https://docs.djangoproject.com/en/4.1/topics/forms/modelforms/>
- [34] Documentation of the django template language. Accessed: 2023-02-17. [Online]. Available: <https://docs.djangoproject.com/en/4.1/ref/templates/language/>
- [35] Bootstrap cards utility documentation. Accessed: 2023-02-25. [Online]. Available: <https://getbootstrap.com/docs/4.0/components/card/>
- [36] Bootstrap modal element documentation. Accessed: 2023-02-27. [Online]. Available: <https://getbootstrap.com/docs/4.0/components/modal/usage>
- [37] About django.contrib.auth package. Accessed: 2023-02-16. [Online]. Available: <https://docs.djangoproject.com/en/4.1/ref/contrib/auth/>
- [38] Documentation of the celery scheduler framework. Accessed: 2023-03-23. [Online]. Available: <https://docs.celeryq.dev/en/stable/getting-started/introduction.html>
- [39] Usage of a sliding window approach in time series prediction with neural network. Accessed: 2023-04-10. [Online]. Available: https://www.ripublication.com/ijcir17/ijcirv13n5_46.pdf

- [40] About time series forecasting problem in supervised machine learning. Accessed: 2023-04-10. [Online]. Available: <https://machinelearningmastery.com/time-series-forecasting-supervised-learning/>
- [41] S. Hochreiter. (1997) Long short term memory. Accessed: 2023-04-17. [Online]. Available: https://www.researchgate.net/publication/13853244_Long_Short-term_Memory
- [42] Summary of the lstm network architecture design. Accessed: 2023-04-17. [Online]. Available: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- [43] Asian research journal of arts & social sciences, july 2018, strategies for increasing student's self-motivation. Accessed: 2022-10-01. [Online]. Available: www.researchgate.net/publication/326556176_Strategies_for_Increasing_Students'_Self-motivation
- [44] A. Vanyukhin. (2021) Application for personal development - bachelor theses structure and inspiration for sources to study from. Accessed: 2022-10-05. [Online]. Available: <https://dspace.cvut.cz/handle/10467/94978>
- [45] Q. V. Tran. (2022) Systém pro sledování osobního rozvoje. Accessed: 2022-10-05. [Online]. Available: <https://dspace.cvut.cz/handle/10467/101722>
- [46] Habits formation. Accessed: 2022-10-11. [Online]. Available: <https://positivepsychology.com/how-habits-are-formed/>
- [47] Comparison of personal project management software tools. Accessed: 2022-10-24. [Online]. Available: <https://monday.com/blog/project-management/project-management-software-for-individuals/>
- [48] List and comparison of personal task management tools. Accessed: 2022-10-24. [Online]. Available: <https://www.proprofsproject.com/blog/personal-task-management-tools/>
- [49] List of personal management tools. Accessed: 2022-10-25. [Online]. Available: <https://timelyapp.com/blog/best-task-management-app>
- [50] How intrinsic motivation is inspired. Accessed: 2022-10-28. [Online]. Available: <https://positivepsychology.com/increase-intrinsic-motivation/>
- [51] About habitnow application. Accessed: 2022-11-05. [Online]. Available: <https://myroomismyoffice.com/habitnow-review/>
- [52] V. Rýdl. (2016) Software pro správu měření. Accessed: 2022-11-24. [Online]. Available: <https://dspace.cvut.cz/handle/10467/65239>
- [53] Functional requirements definition. Accessed: 2022-11-24. [Online]. Available: <https://www.indeed.com/career-advice/career-development/functional-requirements-examples>

- [54] Software architectural patterns types. Accessed: 2022-12-02. [Online]. Available: <https://medium.com/@ankit.sinhal/mvc-mvp-and-mvvm-design-pattern-6e169567bbad>
- [55] List of web application frameworks. Accessed: 2022-12-02. [Online]. Available: <https://www.geeksforgeeks.org/top-10-frameworks-for-web-applications/>
- [56] Installation of bootstrap 5. Accessed: 2023-02-15. [Online]. Available: <https://django-bootstrap-v5.readthedocs.io/en/latest/installation.html>
- [57] R. S. Pressman and B. Maxim, *Software Engineering: A Practitioner's Approach*, 7th ed. McGraw-Hill Education, 2010. [Online]. Available: https://www.mlsu.ac.in/econtents/16_EBOOK-7th_ed_software_engineering_a_practitioners_approach_by_roger_s._pressman_.pdf
- [58] How to send a csrf token via javascript xhr. Accessed: 2023-03-20. [Online]. Available: <https://stackoverflow.com/questions/57516816/cannot-add-csrf-token-to-xmlhttprequest-for-django-post-request>
- [59] Source of user profile default image. Accessed: 2023-03-23. [Online]. Available: https://www.kindpng.com/imgv/JhRRhb_account-user-profile-avatar-avatar-user-profile-icon/
- [60] Installation guide for rabbitmq framework on a macos machine. Accessed: 2023-03-25. [Online]. Available: <https://www.rabbitmq.com/install-homebrew.html>